Marjut Kinnunen

WEB-SOVELLUS JÄTEKESKUKSEN VELVOITETARKKAILUUN

Kajaanin ammattikorkeakoulu
Tradenomikoulutus
Kevät 2006

SISÄLLYS

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Luonnontieteiden ala	Koulutusohjelma Tietojenkäsittely
Tekijä(t) Kinnunen Marjut Anneli	
Työn nimi Web-sovellus jätekeskuksen velvoitetarkkailuun	
Vaihtoehtoiset ammattiopinnot Ohjelmointi	Ohjaaja(t) Haataja Sirpa
	Toimeksiantaja Kainuun jätehuollon kuntayhtymä Eko-Kymppi
Aika kevät 2006	Sivumäärä ja liitteet 31+7

Tämä opinnäytetyö tehtiin Kainuun jätehuollon kuntayhtymä Eko-Kympin (myöhemmin Eko-Kymppi) toimeksiannosta. Toimeksiantona oli kehittää web-sovellus, jolla hallitaan Majasaarenkankaan jätekeskuksen ympäristöluvan mukaista velvoitetarkkailua. Sovelluksen käyttäjiä ovat jätekeskuksen henkilöstö, Eko-Kympin hallinnon henkilöstö ja ympäristövalvonnan viranomaiset.

Jätekeskuksen henkilöstö tallentaa tietokantaan näytteenotot, joita tehdään pohjaveden mittauspisteistä ja vesistömittauspisteistä. Hallinnon henkilöstö tallentaa laboratorion analysoimat tulokset ja hallitsee valvonnan perustietoja sekä tulostaa kaavioita ja taulukoita. Ympäristövalvonnan viranomaiset voivat sovelluksen avulla reaaliaikaisesti seurata velvoitetarkkailun tuloksia.

Sovellus toteutettiin Java 2 Platform Enterprise Edition (J2EE) -tekniikalla, jossa servlettien avulla siirretään ohjelmakoodia html-koodin joukossa asiakkaan selaimelle. Sovelluksen suunnittelussa noudatettiin arkkitehtuuri- ja moduulisuunnittelun periaatteita, jotta sovellus on helposti päivitettävissä ja laajennettavissa.Velvoitetarkkailuun liittyvien mittaustulosten visualisoimiseksi käytettiin JFreeChart-viivakaaviokomponenttia. Sovelluksen tietokantana on Microsoft Access -tietokanta.

Sovelluskehitys jatkuu opinnäytetyön jälkeen, kun sovellusta laajennetaan muihin Kainuun jätehuollon kuntayhtymän ympäristövalvonnan mittauksiin.

Kieli	suomi
Asiasanat	web-sovellus, arkkitehtuurisuunnittelu, velvoitetarkkailu
Säilytyspaikka	☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☒ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School	Degree Programme
Business	Data Processing
Author(s)	
Kinnunen Marjut	
Title	
Web Program for Mandatory Monitoring at a Waste Center	
Optional Professional Studies	Instructor(s)
Programming	Haataja Sirpa
	Commissioned by
	Kainuun jätehuollon kuntayhtymä Eko-Kymppi
Date	Total Number of Pages and Appendices
Spring 2006	31+7

The purpose of the thesis commissioned by Kainuun jätehuollon kuntayhtymä Eko-Kymppi (Joint Municipal Authority for Waste Management in the Kainuu Region) was to build a web program for the waste center of Majasaarenkangas in Kajaani. The program is used to administer mandatory monitoring in accordance with the environmental licence. Users of the program are the personnel in the waste center and in the administration of Eko-Kymppi. The authorities for environmental control can see the real time results of the monitoring.

There are measument points for the groundwater and the water areas in the waste center at Majasaarenkangas. The specimens from these places are taken by the personnel of the center. These operations are entered to the database by the program. The results returned from the laboratory are entered to the database by the administration personnel. Line charts and tables of the measurement results can be obtained.

The program was built by Java 2 Platform Enterprise Edition technique where the program codes are tranferred to clients in html code. The principles of architectural and module planning were followed in the program project to make the program easy to update and expand. A chart component, JFreeChart, was used to show line charts from the measurement results. Microsoft Access was chosen to the database of the program.

The software development will continue after the thesis. A new version of the program will be developed for other measurements of environmental control at Kainuun jätehuollon kuntayhtymä Eko-Kymppi.

Language of Thesis	Finnish
Keywords	websoftware, software architecture, mandatory monitoring
Deposited at	☒ Kaktus Database at University of Applied Sciences Library ☒ Library of University of Applied Sciences

1 JOHDANTO

Opinnäytetyön toimeksiantaja on Kainuun jätehuollon kuntayhtymä Eko-Kymppi (myöhemmin Eko-Kymppi). Se on kymmenen kainuulaisen kunnan jätehuollon viranomaisorganisaatio, joka on aloittanut toimintansa vuonna 2002. Eko-Kympin jäsenkuntia ovat Kajaani, Vuolijoki, Vaala, Ristijärvi, Hyrynsalmi, Puolanka, Paltamo, Sotkamo, Kuhmo ja Suomussalmi. (Eko-Kymppi 2006, 1a)

Eko-Kymppi huolehtii jäsenkuntiensa puolesta jätelaissa kunnan tehtäväksi säädetyn jätehuollon järjestämisestä. Käytännössä tämä tarkoittaa mm. asumisessa syntyvän jätteen keräilyn, kuljetuksen, hyödyntämisen ja käsittelyn järjestämisvastuuta sekä jätehuollon tiedotusta ja neuvontaa. Eko-Kymppi on vastuussa myös jätehuoltoon liittyvistä viranomaistehtävistä. (eml.)

Eko-Kymppi on keskittänyt jätteen vastaanotto- ja käsittelytoiminnot Kajaaniin Majasaarenkankaan jätekeskukseen, jonne on valmistumassa tiukentuvien ympäristölupamääräysten mukainen jätekeskuskokonaisuus vuonna 2007 (Eko-Kymppi 2006, 1a). Vuoden 2007 jälkeen Majasaarenkankaan jätekeskus on ainoa jätteen loppusijoituspaikka Kainuussa. Eko-Kympillä on 20.12.1999 myönnetty ympäristölupa, Dnro 1296Y0091-121, jonka 26. kohdan mukaan kaatopaikalta valuvien vesien määrää ja laatua sekä niiden vaikutuksia pinta- ja pohjaveteen on tarkkailtava (Eko-Kymppi 2006, 1b).

Eko-Kymppi on raportoinut toimintansa alusta alkaen Majasaarenkankaan jätekeskuksen ympäristöluvan mukaisesti velvoitetarkkailun tuloksista ympäristöviranomaisille neljä kertaa vuodessa. Velvoitetarkkailua varten on otettu vesinäytteitä jätekeskuksen pohjaveden mittauspisteistä ja sen ympäristöstä olevista vesistöpisteistä. Velvoitetarkkailun tuloksia on dokumentoitu ainoastaan erilaisilla excel-taulukoilla. Raportoinnin hallinta on ollut monimutkaista ja kankeaa. Velvoitetarkkailuun tehtiin opinnäytteenä web-sovellus, jonka avul-

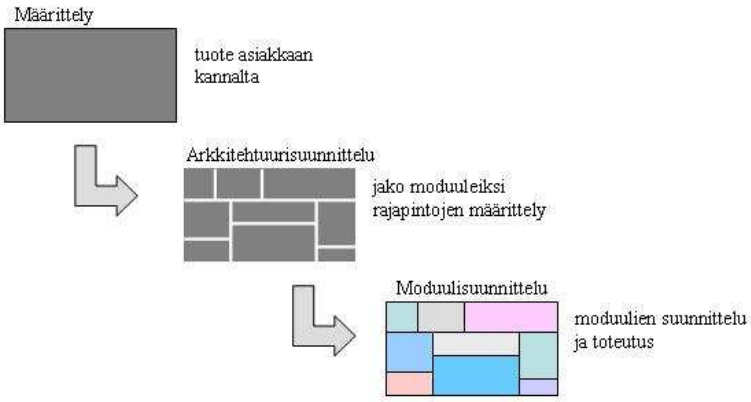
la voidaan tallentaa sähköisesti velvoitetarkkailuun liittyvät näytteenotot ja niiden tulokset sekä tulostaa viivakaavioita ja taulukoita mittaustuloksista.

Opinnäytetyön teoriaosuudessa on käsitelty ohjelmistosuunnitteluun liittyvää arkkitehtuuria. Arkkitehtuurisuunnittelu on keskeisin tekijä onnistuneen sovelluksen kehittämisessä. Arkkitehtuuri- ja moduulisuunnittelun avulla voidaan sovelluskehityksestä tehdä joustavaa ja helposti ylläpidettävää.

Arkkitehtuurimalleja ei ole standardoitu, joten niistä on olemassa paljon erilaisia variaatioita. Arkkitehtuuria voidaan tarkastella monenlaisista näkökulmista. Näistä syistä johtuen ei ole olemassa tyhjentävää vastausta siihen, mitä kaikkea arkkitehtuuri sisältää. Arkkitehtuurin valinta opinnäytteen teoriaosuudeksi oli kiehtovaa ja haastavaa. Opinnäytteeseen liittyvässä teoriaosuudessa tarkastellaan arkkitehtuurisuunnittelua ja -tyylejä rakennearkkitehtuurin näkökulmasta.

2 ARKKITEHTUURISUUNNITTELU

Ohjelmistosuunnittelu voidaan jakaa kolmeen vaiheeseen. Määrittelyvaiheessa tarkastellaan ohjelmistotuotetta asiakkaan näkökulmasta. Arkkitehtuurisuunnittelussa määritellään ohjelmiston tarvitsemat moduulit ja niiden rajapinnat. Moduulisuunnittelussa suunnitellaan ja toteutetaan moduulien tarkempi sisältö. (Kuvio 1.)



Kuvio 1. Ohjelmistosuunnittelun vaiheet (Haikala & Märijärvi 2002, 303)

”Arkkitehtuurisuunnittelu on toteutuksen tärkein vaihe: siinä tehdyt huonot ratkaisut kostautuvat myöhemmin korkeina toteutus- ja ylläpitokustannuksina – hyvässä arkkitehtuurissa yksittäinen huono moduuli on tavallisesti helppo korvata paremmalla.” (Haikala & Märijärvi 2002, 303)

Ohjelmistojen avulla ratkaistaan monimutkaisia ongelmia. Tämän vuoksi ohjelmistojen rakenne voi muotoutua epäyhtenäiseksi ja sekavaksi. Hyvällä arkkitehtuurisuunnittelulla väl-

tytään liian monimutkaisilta rakenteilta ja helpotetaan järjestelmän sidosryhmien välistä kommunikaatiota. Onnistuneen arkkitehtuurisuunnittelun tavoitteena on jakaa ohjelmisto pienempiin ja helpommin hallittaviin kokonaisuuksiin, moduuleihin. Arkkitehtuurisuunnittelun onnistuminen takaa sen, että järjestelmät ovat helposti laajennettavissa ja uudelleenkäytettävissä. (Haikala & Märijärvi 2002, 311)

Ohjelmiston toteutusfilosofia on laadittava yhtenäiseksi, jotta ohjelmistotuotannon lopputuloksena on onnistunut sovellus. Arkkitehtuuri- ja moduulisuunnittelulla ei ole merkitystä, jos niiden työstämisessä ei noudateta yhtenäistä toteutusfilosofiaa, joka käsittää kaikki periaatteet ja rakenteet, joiden ajatellaan pysyvän muuttumattomina koko ohjelmiston elinkaaren ajan. (Haikala & Märijärvi 2002, 314)

2.1 Määritelmiä

Arkkitehtuurisuunnittelussa kuvataan ohjelmistojärjestelmän elementtejä ja niiden sovittamista toisiinsa. Arkkitehtuuri kuvaa myös sitä kuinka elementit toimivat yhdessä ohjelmiston toiminnallisten vaatimusten toteuttamiseksi. Ohjelmistojärjestelmien elementtejä ovat erilaiset ohjelmakomponentit ja niiden väliset rajapinnat. Arkkitehtuuri ei kuitenkaan ole täydellinen ohjelmiston rakennekuvaus. Arkkitehtuurissa määritellään ohjelmiston moduulit ja niiden rajapinnat; moduulien toteutukseen ei oteta kantaa. Moduulien toiminnallisuus kuvataan tarkemmin moduulisuunnittelun vaiheessa. (Helsingin yliopisto, 2002)

Arkkitehtuurisuunnittelu lähtee liikkeelle järjestelmän toiminnallisista perusvaatimuksista, joita kartoitetaan yhdessä ohjelmiston tilaavan asiakkaan kanssa. Kuitenkin laadulliset eli eitoiminnalliset vaatimukset vaikuttavat enemmän tehtäviin rakenneratkaisuihin. Arkkitehtuurin laadullisia vaatimuksia ovat luotettavuus, uudelleenkäytettävyys, skaalautuvuus, muokattavuus, siirrettävyys, testattavuus, tietoturva, vakaus, muistin käyttö ja tehokkuus. (Helsingin yliopisto, 2005)

Arkkitehtuurikuvauksen on oltava kattava, ristiriidaton ja yksiselitteinen, jotta voidaan varmistua ohjelmiston vaatimusten toteutumisesta. Arkkitehtuurin kuvaustekniikoita ovat mm. luokkakaaviot, tilakaaviot, tietovuokaaviot ja muut UML-kuvaukset. (Helsingin yliopisto, 2006)

2.2 Toteutus

Jokaisella järjestelmällä on arkkitehtuuri, olipa sitä suunniteltu tai ei. Onnistuneen lopputuloksen takaamiseksi on välttämätöntä tarkastella arkkitehtuuria jollakin tasolla. Mitä laajempia järjestelmiä ollaan tekemässä, sitä suurempi merkitys on arkkitehtuurisuunnittelulla. Huolellisella arkkitehtuurisuunnittelulla saavutetaan hyvä kommunikointiyhteys ohjelmistojärjestelmän tilaajaan ja saadaan vankka pohja järjestelmän vaatimuksia koskeville neuvotteluille. Suunnitelman avulla pystytään saattamaan järjestelmien vaatimukset realistiselle tasolle. Arkkitehdit, ohjelmoijat ja järjestelmän tilaavat asiakkaat eivät välttämättä ymmärrä järjestelmään liittyviä asioita samalla tavalla. Arkkitehtuurin avulla he pystyvät havainnollisesti keskustelemaan järjestelmän toteutuksesta. (Helsingin yliopisto, 2002)

Arkkitehtuuri on suunniteltava huolellisesti ennen ohjelmistojärjestelmän toteutusvaihetta. Kuitenkin toteutusvaiheessa voi tulla tarve muuttaa arkkitehtuuria. Syitä tähän voivat olla mm. ohjelmistoympäristöjen tai ohjelmistoalustojen muutokset, vääriksi osoittautuneet oletamukset esimerkiksi saavutettavasta suorituskyvystä, kolmansien osapuolien viivästymiset tai hankittavien komponenttien yhteensopimattomuus. Ohjelmistojärjestelmän arkkitehtuurin on kestettävä nämä muutosvaatimukset. Arkkitehtuurin suunnitteluvaiheessa on syytä kartoittaa em. riskitekijöitä, jotta arkkitehtuuria voidaan tarvittaessa muokata uusien vaatimusten mukaiseksi. (Helsingin yliopisto, 2005)

Ohjelmiston tekijät vaihtuvat, mutta ohjelmiston arkkitehtuuri pysyy. Jatkuvuuden takaamiseksi on erittäin tärkeää, että arkkitehtuuri kuvataan riittävän tarkasti. Näin voidaan taata, että ohjelmistokehitys jatkuu tekijöiden vaihtuessakin alkuperäisten vaatimusten mukaisesti. Arkkitehtuurisuunnitelma on laadittava kirjallisena ja kuvakset tehtävä yhteisesti hyväksytyillä tavoilla, jotka kaikki tekijät ymmärtävät määrittelyt samalla tavalla. (Haikala & Märijärvi 2002, 311)

Ohjelmistokehitysprojekteissa vastuu arkkitehtuurin kokonaishallinnasta on aina annettava yhdelle henkilölle. Näin varmistetaan, että arkkitehtuuri säilyy ohjelmistokehityksen ajan asetettujen määrittelyjen mukaisena. Etenkin laajemmissa ohjelmistokehitysprojekteissa vastuuhenkilön on oltava kokenut ohjelmistokehittäjä ja hänen on hallittava arkkitehtuurin kuvaustekniikoita. Vastuuhenkilön on tunnettava organisaatio, joka tekee ohjelmistokehitystyön, jotta ohjelmistoarkkitehtuuri palvelee organisaation toimintatapoja, tavoitteita ja ohjelmistokehityksen asettamia vaatimuksia. Vastuuhenkilön on ymmärrettävä ohjelmistotuotteen kokonaisvaatimukset ja rajoitteet. Arkkitehtuurisuunnitelman avulla vastuuhenkilö pys-

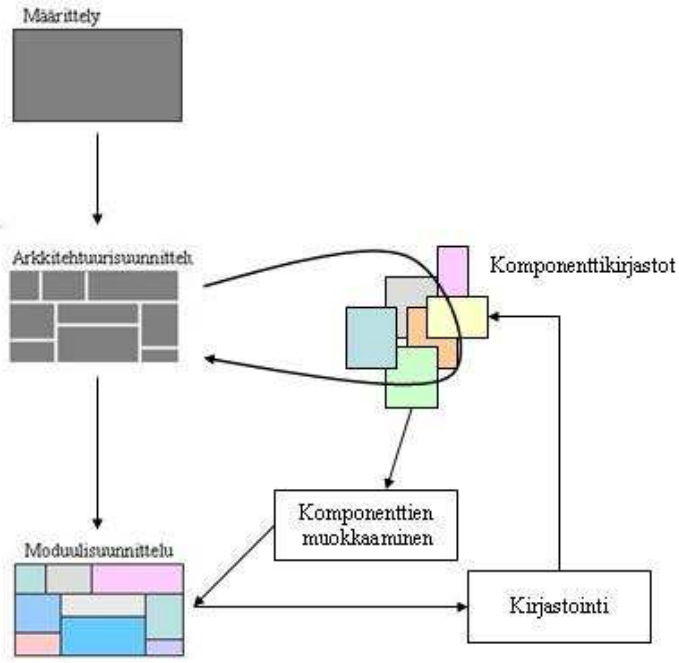
tyy määrittelemään mitä organisaatio tuottaa itse ja mitkä elementit ostetaan organisaation ulkopuolelta. (Helsingin yliopisto, 2005)

2.3 Huonon arkkitehtuurin vaikutukset

Arkkitehtuurin avulla tarjotaan korkean tason näkemys ohjelmistoista. Tästä huolimatta arkkitehtuuri on rakennettava riittävällä tarkkuudella. Muutoin esille voi tulla ongelmia aikataulutuksessa, järjestelmän ylläpidossa tai asiakkaan vaatimusten täyttymättömyydessä. Arkkitehtuuri on voitu suunnitella liian monimutkaiseksi tai liian joustavaksi, jolloin ohjelmiston ositusta ei voida korjata kesken ohjelmistokehityksen. Tällöin ohjelmiston toimitus voi myöhästyä tai koko ohjelmisto voi osoittautua jopa toteutuskelvottomaksi. Jos arkkitehtuuriin ei ole kiinnitetty riittävästi huomioita voi testausvaiheessa selvitä, että jotakin asiakkaan vaatimusta ei pystytä toteuttamaan halutulla tavalla. Tämä voi aiheuttaa ongelmia aikataulutuksessa ja ohjelmiston virheiden hallitussa korjaamisessa. Huonosti laaditun arkkitehtuurin seurauksena ohjelmiston uuden version tuottaminen vaatii paljon vaivaa ja koodin uudelleenkirjoittamista sekä raskaiden rinnakkaisten toimintojen rakentamista. (Haikala & Märijärvi 2002, 311)

2.4 Modulointi

Arkkitehtuurissa määritellään moduuleita, jotka sisältävät järjestelmän toiminnallisuuden. Toiminnallisuuden pilkkominen pienempiin osiin selkeyttää ohjelmistojen kehitystyötä. Ohjelmistoon voidaan myöhemmin lisätä esimerkiksi uusia kirjastoja, joiden avulla saadaan käyttöön uusia komponentteja (Kuvio 2.). Moduloinnin avulla voidaan ohjelmistoon helposti liittää myös uusia palvelimia, jotka tarjoavat uusia palveluita järjestelmän käyttöön. Kun ohjelmiston toiminnot on jaettu pieniin osiin, voidaan näitä osia käyttää uudelleen saman järjestelmän muihin moduuleihin tai hyödyntää näitä osia kokonaan toisessa yhteydessä, esimerkiksi toisessa järjestelmässä. Näin varmistetaan, että kerran tehtyä asiaa ei tarvitse tehdä uudestaan vaan se voidaan rajapintoja käyttämällä kutsua mistä tahansa järjestelmän osasta halutun moduulin käyttöön. Harkitulla moduloinnilla järjestelmät ovat helposti ylläpidettävissä. (Haikala & Märijärvi 2002, 311)



Kuvio 2. Uudelleenkäyttö ja laajentaminen tuotantoprosessissa (Haikala & Märijärvi 2002, 328)

2.5 Abstraktiot

Onnistunut ohjelmiston osittaminen perustuu abstraktioiden käyttöön. Abstraktio on malli, jossa kuvataan esitetystä asiasta vain oleellinen, epäoleelliset asiat jätetään vaille huomioita. Abstraktioiden hahmottaminen helpottaa arkkitehtuuri- ja moduulisuunnittelua. Niiden avulla löydetään olennaiset osiot ohjelmistotuotantoon. Abstraktioilla voidaan jakaa suunnitteluongelmia pienenpiin osiin, jotka voidaan ratkaista itsenäisesti. (Haikala & Märijärvi 2002, 311 - 313)

Abstraktio voi olla esimerkiksi luokka, joka palauttaa määrittelemiään attribuutteja toiselle luokalle. Palautettavat attribuutit sisältyvät abstraktioon. Luokassa voi olla sisäisiä metodeja, jotka muotoilevat luokan sisäisessä käytössä olevien attribuuttien avulla palautettavaa attribuuttia. Nämä sisäiset attribuutit eivät kuulu abstraktioon, sillä ne ovat epäolennaisia luokan ominaisuuksia. Abstraktioksi voidaan määritellä myös useita luokkia sisältävä kokonaisuus. (eml.)

Arkkitehtuuri- ja moduulisuunnittelussa käytetään hyväksi abstraktioita. Sovelluskerros tai moduuli näkee abstraktiosta vain sen rajapinnan, jonka sisään on koteloitu tarvittavat toteu-

tukset. Abstraktiot ovat juuri tästä syystä hyödyllisiä. Kun rajapinnat ovat selkeitä, voidaan toteutuksen yksityiskohdat pilkkoa pienempiin osiin abstraktioiden sisään eikä toteutuksia tarvitse tehdä useassa paikassa. Abstraktion käytöllä saavutetaan etua muunneltavuuteen, sillä hyvin suunnitellun abstraktion toteutuksen voi vaihtaa, jos rajapintaa ei muuteta. (Haikala & Märijärvi 2002, 311 - 313)

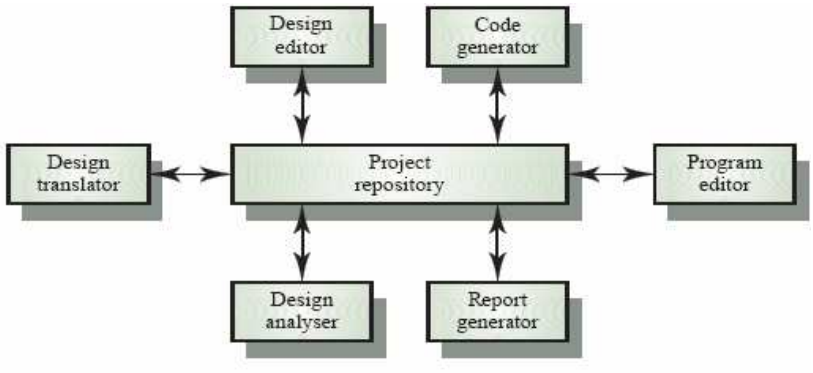
2.6 Arkkitehtuurityylit

Arkkitehtuurisuunnittelu noudattaa tiettyä arkkitehtuurityyliä. Arkkitehtuurityyli kuvaa tapaa, jolla järjestelmän elementit liitetään toisiinsa. Tyyli voi asettaa rajoituksia elementtien toisiinsa liittämiseen. Tyylejä voidaan myös yhdistellä esimerkiksi siten, että korkeamman tason suunnittelu noudattaa erilaista tyyliä kuin alemman tason suunnittelu.

Arkkitehtuurityyleistä esitetään useita malleja. Mallit vaihtelevat sen mukaan, mistä näkökulmasta niitä tarkastellaan. Tarkastelunäkökulmana voi olla esimerkiksi ohjelmiston rakenne tai hajautettavuus. Opinnäytteessä on käsitelty tunnetuimpia rakennearkkitehtuurityylejä, joita ovat tietovarastokeskeinen arkkitehtuuri, client-server -arkkitehtuuri, tietovuoarkkitehtuuri ja kerrosarkkitehtuuri.

2.6.1 Tietovarastokeskeinen arkkitehtuuri

Tietovarastokeskeisessä arkkitehtuurissa tietovarasto on keskeisin osa ohjelmistojärjestelmää. Kaikki ohjelmakomponenttien toiminnot ohjautuvat tietovaraston kautta (Kuvio 3.). Komponentit päivittävät, hakevat, lisäävät ja poistavat tietovaraston tietoja. Komponentteja voidaan muokata, lisätä tai poistaa ilman, että sillä on järjestelmän muihin komponentteihin vaikutusta. Tietovarastokeskeinen arkkitehtuuriratkaisu on hyvä siinä tapauksessa, kun jaetaan suuria määriä tietoa. Tällöin ei tarvitse suunnitella alijärjestelmiä tiedon jakamiseen vaan kaikki järjestelmän komponentit ovat yhteydessä tietovarastoon saman mallin mukaisesti. Toisaalta tämä aiheuttaa sen, että komponenttien on oltava tältä osin identtisiä eikä kaikkia haluttuja komponentteja saada kytkettyä järjestelmään yhteensopivuusongelmien vuoksi. (Turun yliopisto, 2002)

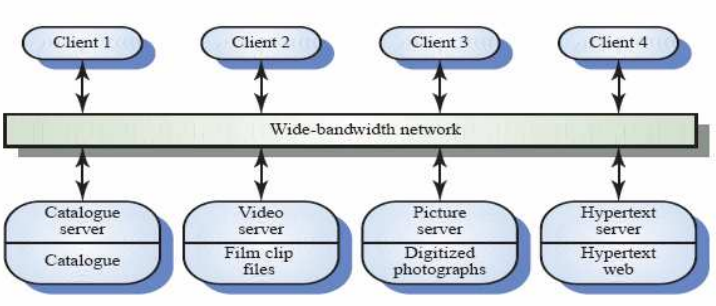


Kuvio 3. Tietovarastokeskeisen arkkitehtuurin kuvaus (eml.)

Tietovarastokeskeisen arkkitehtuurin suurin ongelma on hajauttamisen vaikeus. Jos tietovarastoa halutaan hajauttaa esimerkiksi useammalle palvelimelle, on kaikki ohjelmistojärjestelmän komponentit päivitettävä. Tämän seurauksena myös ylläpidettävyys- ja uudelleenkäytettävyysominaisuudet heikkenevät. (eml.)

2.6.2 Client-server -arkkitehtuuri

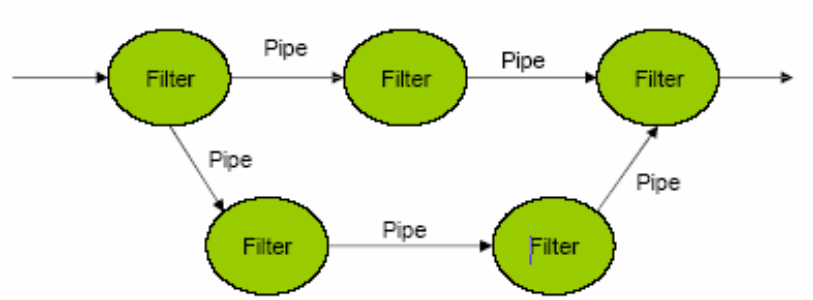
Client-server -arkkitehtuuri kuvaa tiedon ja sen prosessoinnin jakamista ohjelmistojärjestelmän komponenteille. Arkkitehtuurin pääkomponentteja ovat palvelin, asiakas ja verkko (Kuvio 4.). Palvelin tuottaa palveluita, joita asiakaskomponentit käyttävät verkon välityksellä. Client-server -arkkitehtuuriin on helppo lisätä palvelin- ja asiakaskomponentteja. Asiakas- ja palvelinkomponenttien yhteydet hoidetaan verkon välityksellä, joten yhteyskäytännöt on määriteltävä huolellisesti. (Turun yliopisto, 2002)



Kuvio 4. Client-server -arkkitehtuurin kuvaus (eml.)

2.6.3 Tietovuoarkkitehtuuri

Tietovuoarkkitehtuuri sopii järjestelmiin, joissa tietovirran jalostaminen ja prosessointi on olennaista. Arkkitehtuurin komponentteja ovat prosessointiyksiköt ja väylät. Prosessointiyksiköt käsittelevät tietoa ja väylät siirtävät tietoa eteenpäin (Kuvio 5.). Väylien toiminta on passiivista, ne tarvitsevat impulssin tiedon siirtämisestä prosessointiyksiköltä. Prosessointiyksiköt voivat toimia täysin itsenäisesi. Ne lukevat syötevirtaa ja tekevät syötteiden perusteella toimenpiteitä yksikön sisällä. Toimenpiteiden jälkeen prosessointiyksikkö lähettää muokatun tiedon eteenpäin väylää pitkin. (Helsingin yliopisto, 2006 1b.)



Kuvio 5. Tietovuoarkkitehtuurin kuvaus (eml.)

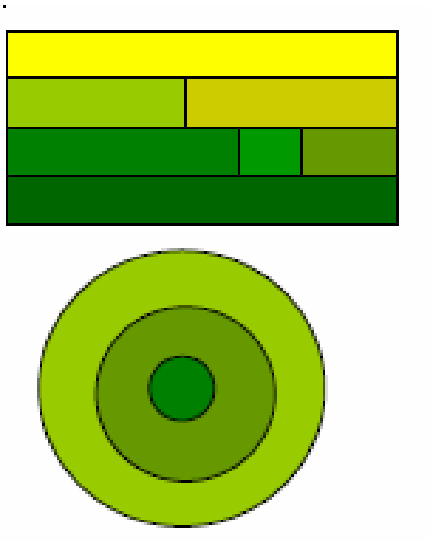
Tietovuoarkkitehtuuri on suoraviivainen arkkitehtuurityyli. Jokainen prosessointiyksikkö voi toimia omassa säikeessään. Vaativat toimenpiteet voidaan eritellä omaan prosessointiyksikköön esimerkiksi omille palvelimilleen, jolloin ohjelmistojärjestelmien suorituskykyä voidaan merkittävästi parantaa. Tietovuoarkkitehtuuri käsittelee tietoa vaiheittain. Monimutkaisia prosesseja voidaan toteuttaa askel askeleelta, jolloin varmistetaan tiedon oikeellisuus. Toisaalta vaiheittaisessa prosessoinnissa on riskinä alkuperäisen tiedon muuttuminen prosessoinnin aikana niin, että prosessointiketjun seuraavat prosessointiyksiköt eivät vastaanotakaan tietoa vaaditussa muodossa. Tietovuoarkkitehtuurissa virheenkäsittely on hoidettava huolellisesti, jotta em. kaltaiselta virheeltä voidaan suojautua. (eml.)

Tietovuoarkkitehtuuri ei sovi käytettäväksi interaktiivisissa järjestelmissä, sillä prosessointiketjun aikana ei ole sellaista järjestelmän tilaa, joka voitaisiin järkevällä tavalla esittää käyttäjälle esimerkiksi käyttöliittymän avulla. Tietojen salausohjelmistot käyttävät usein

tietovuoarkkitehtuuria. Salausohjelmissa käyttäjälle ei näytetä tiedon salausprosessia lainkaan vaan se tapahtuu täysin ohjelmaprosessien sisällä. (eml.)

2.6.4 Kerrosarkkitehtuuri

Kerrosarkkitehtuurissa järjestelmä jaetaan yhtenäisiin käsitekerroksiin. Arkkitehtuuri koostuu tasoista, jotka on järjestetty abstraktioiden määrittelyiden perusteella nousevaan järjestykseen. Kerrosarkkitehtuuria voidaan kuvata vaakakerroksina tai sisäkkäisinä kerroksina (Kuvio 6.) Ylempi tai ulompi kerros kutsuu alemman tai sisemmän kerroksen palveluita. Kutsuja voidaan tehdä toiseenkin suuntaan, mutta sitä on kuitenkin syytä välttää, sillä yleensä alempi kerros on riippuvainen ylemmästä kerroksesta. Palvelukutsun tekeminen alemmasta kerroksesta ylempään voi olla merkki ohjelmiston huonosta arkkitehtuurista. (Helsingin yliopisto, 2006 1b.)

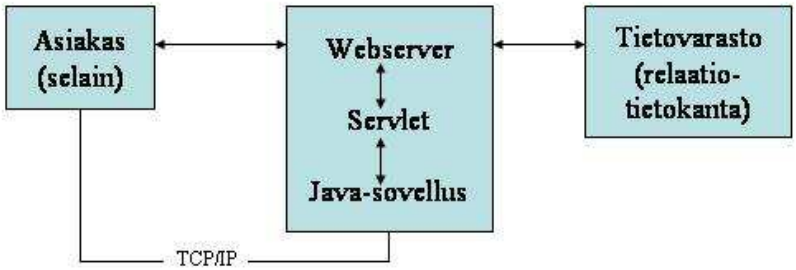


Kuvio 6. Kerrosarkkitehtuurin kuvaus vaaka- tai sisäkkäisrakenteella (eml.)

Kerrosarkkitehtuuri on rakenteena helposti ymmärrettävissä, siksi sitä voidaan käyttää kommunikoinnissa hyvinkin erilaisten sidosryhmien välillä. Se soveltuu hyvin isossa tai pienessä mittakaavassa toteutettaviin ohjelmistojärjestelmiin. Kerrokset ovat selkeästi omina kokonaisuuksinaan, joten niiden ylläpitäminen ja laajentaminen on helppoa. Kerrosarkkitehtuuri on hyvä ratkaisu silloin, kun ohjelmistojärjestelmissä tavoitellaan uudelleenkäytettävyyttä. Ohjelmistokehitystyössä voidaan hyödyntää tasokohtaista erikoistumista työnjaossa. Esimerkiksi ohjelmistokehitystyössä olevilla työntekijöillä vain osalla on osaamista alemman kerroksen tietokannoista ja osalla ylemmän kerroksen käyttöliittymätoiminnoista. (eml.)

2.7 Kolmikerroksiset web-sovellukset

Web-sovellukset jaotellaan arkkitehtuurityyliltään kerrosarkkitehtuuriin. Ensimmäisessä kerroksessa on asiakas eli selain, toisessa kerroksessa web-palvelin ja kolmannessa tietovarasto eli relaatiotietokanta (Kuvio 7.). Ensimmäisen ja toisen kerroksen väliset yhteydet hoidetaan TCP/IP-verkon avulla. Toinen ja kolmas kerros yhdistetään tietokantayhteyssillan avulla. (Nakhimovsky & Myers 2002, 32 – 33)



Kuvio 7. Hajautettu kolmikerroksinen web-sovellus (Nakhimovsky & Myers 2002, 33)

Web-sovellukset ovat hajautettuja järjestelmiä. Hajautettu web-sovellus jaetaan arkkitehtuurin osalta kolmeen kerrokseen. Hajautetuissa järjestelmissä arkkitehtuurisuunnittelulla on keskeinen rooli. Hajautetussa järjestelmässä joukko itsenäisiä tietokoneita on kytketty toimimaan yhdessä niin, että käyttäjän näkökulmasta katsottuna se vaikuttaa yhdeltä kiinteältä järjestelmältä. Hajautetut järjestelmät perustuvat kuvion 1 mukaiseen periaatteeseen, jossa koko ohjelmisto ositetaan siten, että jokaista ohjelmiston osaa voidaan kuvata ja kehittää ohjelmistotason työvälineillä. Järjestelmien hajauttamisen tavoitteena on rakentaa avoin järjestelmä, jossa järjestelmän palveluita voidaan käyttää huolellisesti määriteltyjen rajapintojen kautta mistä tahansa järjestelmän osasta. Hajauttamisella pyritään myös siihen, että järjestelmä on helposti laajennettavissa. (Hatakka Taina, 2004)

Web-sovelluksen kolmikerroksinen arkkitehtuuri luo turvallisuutta sovelluksen toimintaan. Mikäli yhteen kerrokseen tulee ongelmia, ei koko järjestelmää tarvitse korjata vaan voidaan turvallisesti tehdä korjauksia vain yhden kerroksen toimintoihin. Kun arkkitehtuurin kukin kerros vielä jaetaan pienempiin osiin moduulisuunnittelun avulla, saadaan järjestelmästä kokonaisuutena tehokas, luotettava, uudelleenkäytettävä, laajennettava ja siirrettävä. (Haikala & Märijärvi 2002, 308)

Ensimmäinen sovelluskerros

Kolmikerroksinen arkkitehtuuri tekee web-sovelluksista ensimmäisen kerroksen osalta järjestelmäriippumattomia. Sovellusten käyttöliittymät esitetään hmtl-kielellä. Tämä mahdollistaa sen, ettei web-sovelluksen käyttöönotto asiakkaalla välttämättä vaadi erillisasennuksia, sillä lähes jokaisessa tietokoneessa on nykyisin asennettuna jokin selain. Lisäksi saavutetaan etua siitä, että useat käyttäjät voivat käyttää palvelimella olevaa web-sovellusta samanaikaisesti. Kolmikerroksinen arkkitehtuuriratkaisun avulla saadaan järjestelmään enemmän suorituskykyä, kun ohjelmistokomponenttien toiminnot voidaan suorittaa web-palvelimella. Sovelluksen toiminnot eivät ole näin ollen riippuvaisia asiakaskoneen suorituskyvystä.

Toinen sovelluskerros

Toiseen web-sovelluksen arkkitehtuurikerrokseen kuuluu web-palvelin. Palvelimelle asennetaan varsinainen ohjelmistokomponentti eli esimerkiksi java-sovellus, joka voidaan toteuttaa servlet-luokkien avulla. Servletit ovat pieniä palvelimella toimivia ohjelmia. Ne eivät kuitenkaan ole itsenäisiä ohjelmia vaan tarvitsevat toimiakseen java-sovelluksen, joka sisältää ohjelman käynnistävän main-metodin, sillä servleteillä ei ole omaa main-metodia. (Nakhimovsky & Myers 2002, 32 – 33, 36)

Servletit ovat turvallinen arkkitehtuuriratkaisu. Niiden avulla liitetään ohjelmakoodia hmtl-koodin joukkoon. Servlet kutsuu web-palvelimella olevia ohjelmaluokkia, joissa ohjelman toiminnallisuus on toteutettu. Servletin avulla voidaan prosessoitu tieto välittää edelleen asiakaskoneelle hmtl-koodissa. Näin ollen asiakaskone ei pääse lainkaan käsiksi toiminnallisiin ohjelmaluokkiin vaan servlet tarjoaa ainoastaan rajapinnan ohjelmaluokkiin.

Kolmas sovelluskerros

Kolmannessa web-sovelluksen arkkitehtuurikerroksessa on tietovarasto, esimerkiksi relaatiotietokanta. Kun tietovarasto suunnitellaan omaan arkkitehtuurikerrokseen, voidaan tietokanta asentaa omalle palvelimelle. Toiseen kerrokseen rakennetaan rajapinta, jonka avulla sovellukset pitävät yhteyttä tietokantaan. Tällöin tietokantaa voidaan ylläpitää toisesta kerroksesta riippumatta ja tietokanta voidaan helposti liittää myös toiseen järjestelmään.

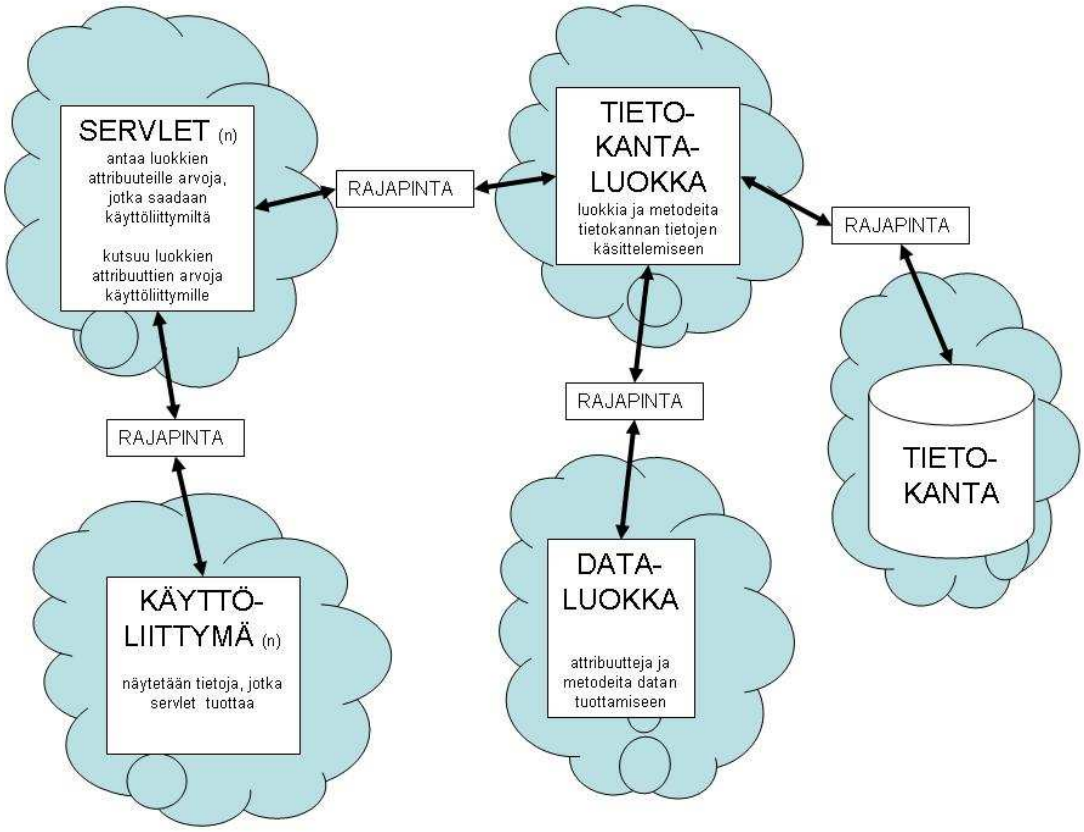
3 WEB-SOVELLUS

Opinnäytteenä tehty sovellus on laadittu kerrosarkkitehtuurityylin mukaisesti. Arkkitehtuuri noudattaa kolmikerroksisen web-sovelluksen periaatteita. Kolmikerroksisuus takaa sen, että sovellus on helposti päivitettävissä. Sovellus koostuu kolmesta tasosta. Ensimmäistä sovelluksen tasoa vastaavat käyttäjien selaimet. Toisessa tasossa ovat palvelimelle asennettu sovellus ja kolmannessa palvelimelle asennettu Access-tietokanta.

Kuten aiemmin esitettiin, täydellisessä kolmikerroksisessa web-sovelluksessa tietovarasto eli tietokanta on omassa kerroksessaan, jopa omalla palvelimellaan. Opinnäytetyönä tehty sovellus on kuitenkin niin suppea, että tietokanta on suunniteltu ylläpidettäväksi samalla palvelimella java-sovelluksen kanssa.

Sovellukseen on lisätty ohjelmistokehityksen aikana ohjelmakirjasto kaaviokomponentin käyttämiseksi. Kaikki ohjelman toiminnalliset kokonaisuudet on moduloitu omiin luokkiinsa moduulisuunnittelun periaatteen mukaisesti. Niinpä uuden ohjelmakirjaston liittäminen sovellukseen ei ole aiheuttanut mitään muutoksia muihin ohjelmaluokkiin. Kaaviokomponentti rakennettiin myös omaan moduuliinsa.

Sovelluksessa on määritelty abstraktioita, joita ovat yksittäiset servletit, käyttöliittymät, tietokantaluokat, muut luokat ja tietovarasto. (Kuvio 8.)



Kuvio 8. Sovelluksen kattavat abstraktiot

3.1 Lähtötilanteen kartoitus

Majasaarenkankaan jätekeskuksessa on vedenmittauspiste, johon jätekeskusten vedenmittauspisteiden mittaustekniikoihin erikoistunut yritys, EHP-tekniikka, on asentanut sähköiset mittauslaitteet. Näiden mittauslaitteiden tuottamat mittaustulokset ovat Eko-Kympin käyttäjätunnuksilla luettavissa EHP-tekniikan Internet-sivuilla. EHP-tekniikan mittauslaitteet on asennettu vain yhteen mittauspisteeseen. (EHP-tekniikka 2006)

Eko-Kympillä on jätekeskuksessaan useita vedenmittauspisteitä, jotka kuuluvat velvoitetarkkailun piiriin, mutta niissä ei ole sähköisiä EHP-tekniikan ylläpitämiä mittauslaitteita. Näistä mittauspisteistä otetaan velvoitetarkkailua varten vesinäytteitä manuaalisesti. Näytteet lähetetään tutkittavaksi asianmukaiseen laboratorioon, joka lähettää analysoinnin jälkeen mittausten tulokset Eko-Kympille.

Eko-Kympillä haluttiin kaikista vedenmittauspisteistä samantyyppisiä kaavioita kuin EHP-tekniikan tarjoamassa mittausjärjestelmässä on. Velvoitetarkkailuun haluttiin sovellus, jolla

ko. kaavioita voi tulostaa. Eko-Kympillä päätettiin tarjota mahdollisuus toteuttaa velvoitetarkkailuun liittyvä sovellus opinnäytetyönä.

Sovelluksen käyttäjiä ovat jätekeskuksen henkilöstö, joka kirjaa tietoja tietokantaan syöttölomakkeen avulla, ja hallinnon henkilöstö, joka tallentaa laboratorion analyysitulokset tietokantaan ja tulostaa tietokannasta erilaisia raportteja ja ylläpitää tietokantaa. Sovellusta käyttävät myös ympäristövalvonnan viranomaiset, joille annetaan sovellukseen selailukäyttöoikeudet. Sovellus liitetään myöhemmin osaksi Eko-Kympin kotisivuja siten, että Internetin käyttäjät voi katsella mittauksista tehtyjä kaavioita omilla selaimillaan.

Ennen web-sovelluksen suunnittelun aloittamista kartoitettiin velvoitetarkkailun lähtötilanne ja tarkkailuun liittyvät vaatimukset. Tavoitteena oli kehittää sovellus, jonka avulla pystytään vaivattomasti kirjaamaan näytteenotot ja mittausten tulokset sekä tulostamaan mittauksista visuaalisia raportteja. Koska toimijoita velvoitetarkkailuketjussa on useita; näytteenottajat, laboratorio, hallinnon henkilöstö ja ympäristövalvonnan viranomaiset, päädyttiin web-pohjaiseen sovellukseen. Tällöin sovellus on mahdollisimman helposti kaikkien toimijoiden käytettävissä eikä erillisiä asennuksia tarvitse toimijoiden koneille tehdä. Kaikki toimijat pystyvät käyttämään sovellusta selaimiensa avulla.

3.2 Suunnittelu

Suunnitteluvaiheessa laadittiin ohjelmakuvaus, jossa esitettiin sovelluksen yleiskuvaus, toiminnalliset vaatimukset, käyttöliittymät, mallinnukset ja määriteltiin sovelluksen tietokantamalli. Kun ohjelmakuvaus oli Eko-Kympillä hyväksytty, aloitettiin sovelluksen toteutusvaihe.

Mittaustuloksia kerätään kahdentyyppisistä vedenmittauspisteistä: pohjaveden mittauspisteistä ja vesistöpisteistä. Pistetyypeistä kerätään hieman erilaisia mittaustuloksia, joten lähtökohtana oli, että molemmille pistetyypeille tehdään omat toiminnot tulosten tallennukseen ja raportointiin.

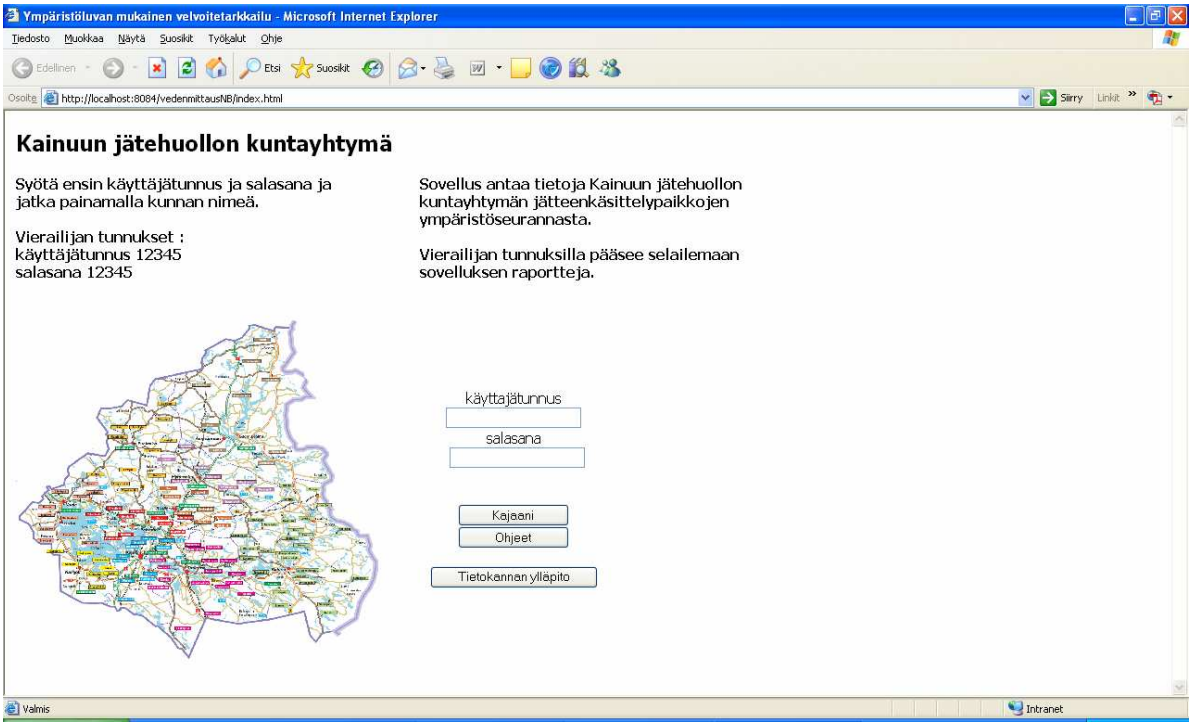
3.3 Toteutus

Eko-Kympillä ei ole sovelluskehitykseen tarvittavia työkaluja. Eko-Kymppi ei asettanut kehitysympäristölle erityisvaatimuksia vaan tavoitteena oli saada toimiva sovellus velvoite-

tarkkailun toteuttamiseen millä kehitystyökalulla tai ohjelmointikielellä tahansa. Toteutustyökalut sai siis valita vapaasti. Sovelluskehittimeksi valittiin Internetistä saatava maksuton Netbeans 4.0, jossa kehitystyökaluna oli Java(TM)2 SDK, Standard Edition versio 1.4.2. Web-sovelluksen ohjelmointikieli oli siis java. Web-sovellus on laadittu Java 2 Platform Enterprise Edition (J2EE) -tekniikalla, jossa hmtl-sivujen sisällön tuottamiseen käytetään servlettejä.

3.4 Sovelluksen käyttöliittymät ja niihin liittyvät toiminnot

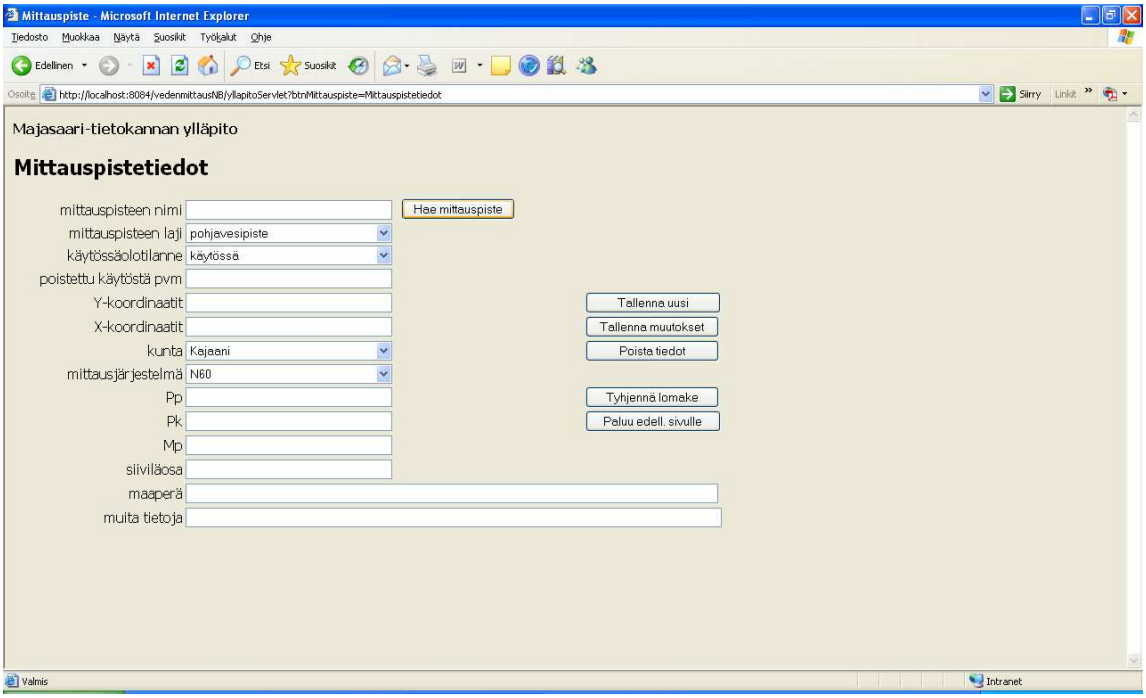
Sovelluksen toiminnoista laadittiin käyttötapauskaavio kaikkien tarvittavien toimintojen kartoittamiseksi (Liite 1.). Sovelluksen käynnistyessä avautuu index-sivu, jossa käyttäjän on kirjauduttava sovellukseen (Kuvio 9.).



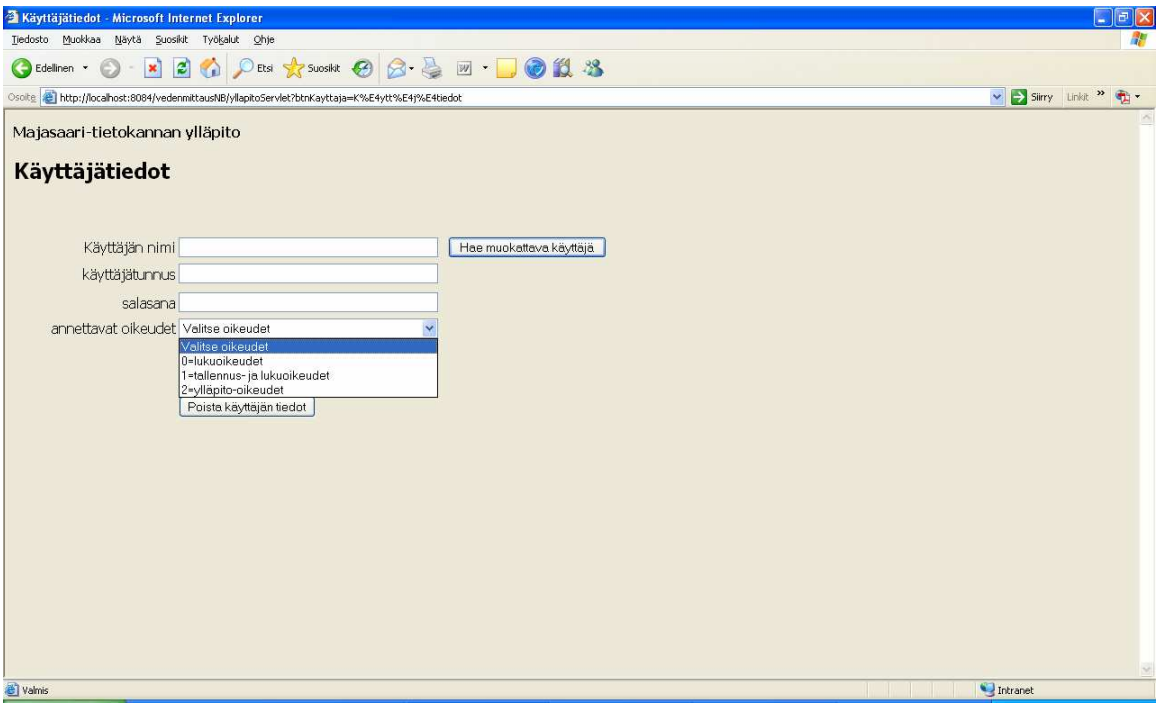
Kuvio 9. Sovelluksen aloitussivu

Sovellus haarautuu aloitusnäytöltä kahteen osioon, joista toinen sisältää ylläpitoon liittyvät toiminnot ja toinen mittauksiin liittyvät tallennukset ja raportit. Osioihin pääsyä on rajattu käyttöoikeuksilla, jotka määritellään käyttäjätietojen tallentamisen yhteydessä. Sovellukseen on määritelty kolmenlaisia käyttöoikeuksia. Ylläpito-oikeuksilla käyttäjä pääsee mihin tahansa sovelluksen osioon, luku- ja kirjoitusoikeuksilla vain mittaustulosten tallennus- ja raportointiosioon ja lukuoikeuksilla mittaustulosten raportointiosioon.

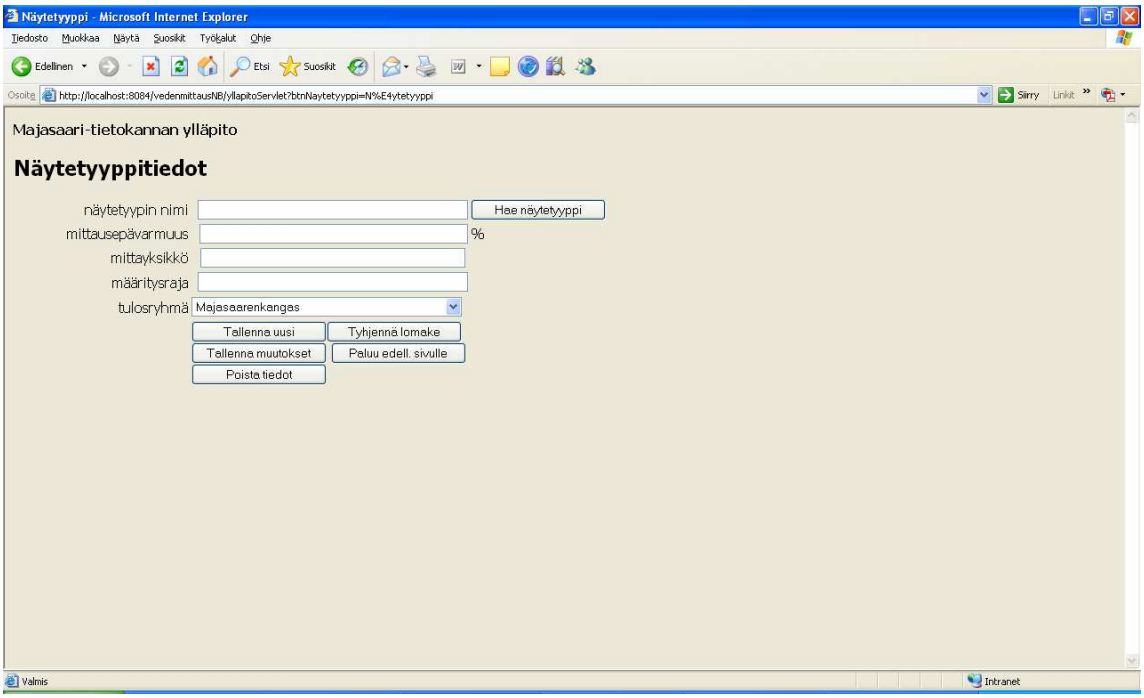
Sovelluksen ylläpito-osioon siirrytään aloitussivun Tietokannan ylläpito -painikkeella. Ylläpito-osiossa tallennetaan yksilöintitietoja vedenmittauspisteistä, analyysituloksista ja sovelluksen käyttäjistä. Ylläpito-osioon pääsevät vain ne käyttäjät, joille on määritelty ylläpito-oikeudet. Ylläpitäjä voi tallentaa mittauspisteestä yksityiskohtaisia tietoja mm. sijainnista ja ominaisuuksista (Kuvio 10). Käyttäjätiedoissa yksilöidään käyttäjä ja määritellään käyttäjätunnus ja salasana sekä käyttöoikeuden taso (Kuvio 11.). Analyysituloksista voidaan tallentaa nimi, määritysraja, mittausepävarmuus ja mittayksikkö (Kuvio 12.).



Kuvio 10. Mittauspistetietojen tallennus -käyttöliittymä

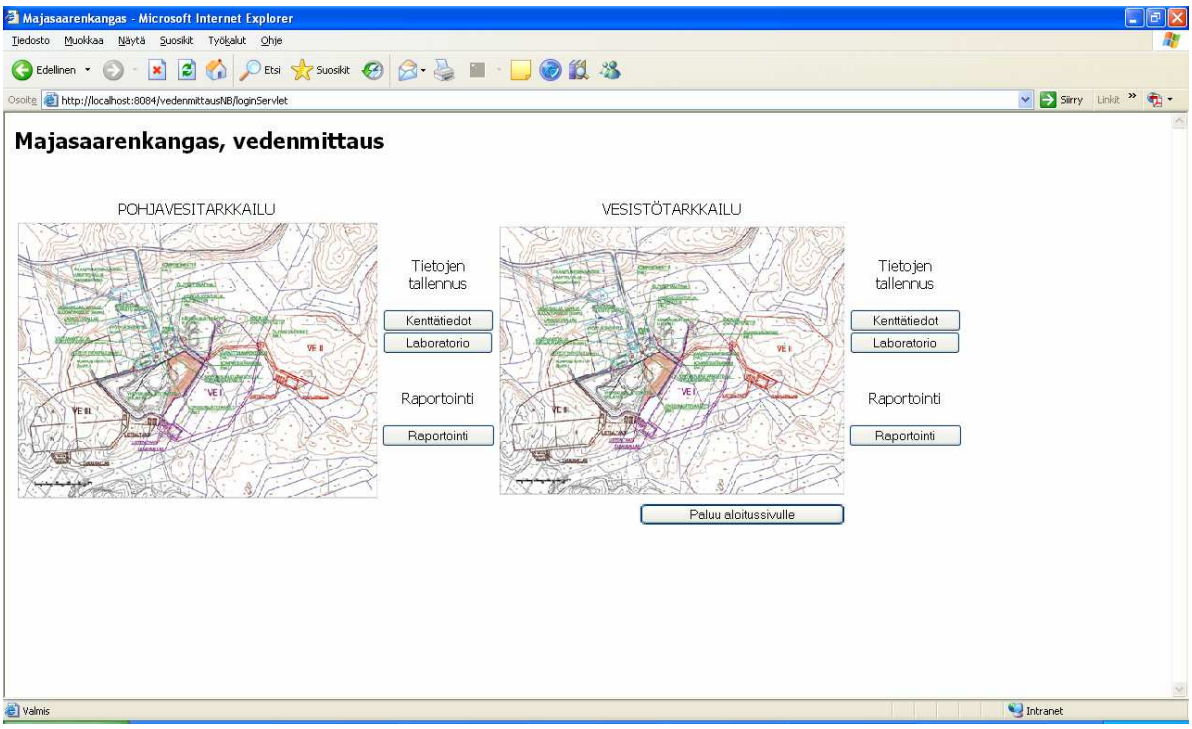


Kuvio 11. Käyttäjätietojen tallennus –käyttöliittymä



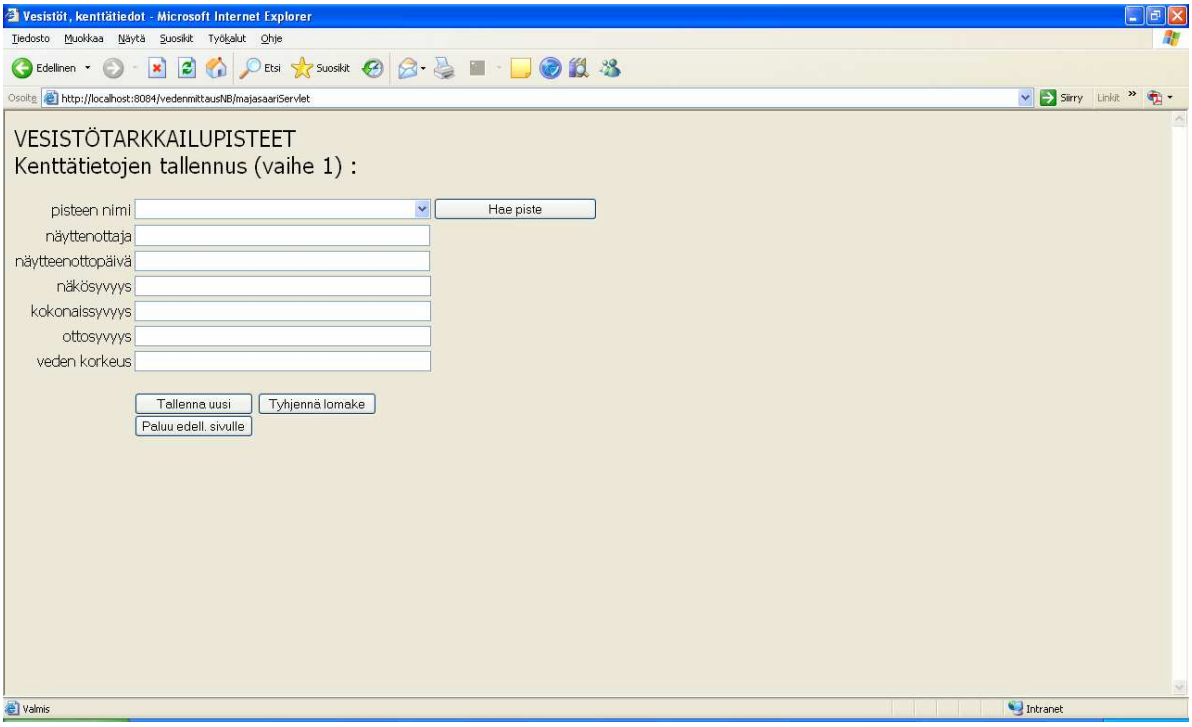
Kuvio 12. Analyysituloksiin liittyvien tietojen tallennus -käyttöliittymä

Aloitussivulla käyttäjän antaessa käyttäjätunnuksen ja salasanan sekä painaessa Kajaani-painiketta päästään sovelluksen osioon, josta edetään näytteenottojen ja laboratoriotulosten tallentamiseen ja korjaamiseen sekä raportointiin (Kuvio 13.). Pohjaveden mittauspisteet ja vesistöpisteet on eritelty omiksi osioikseen.



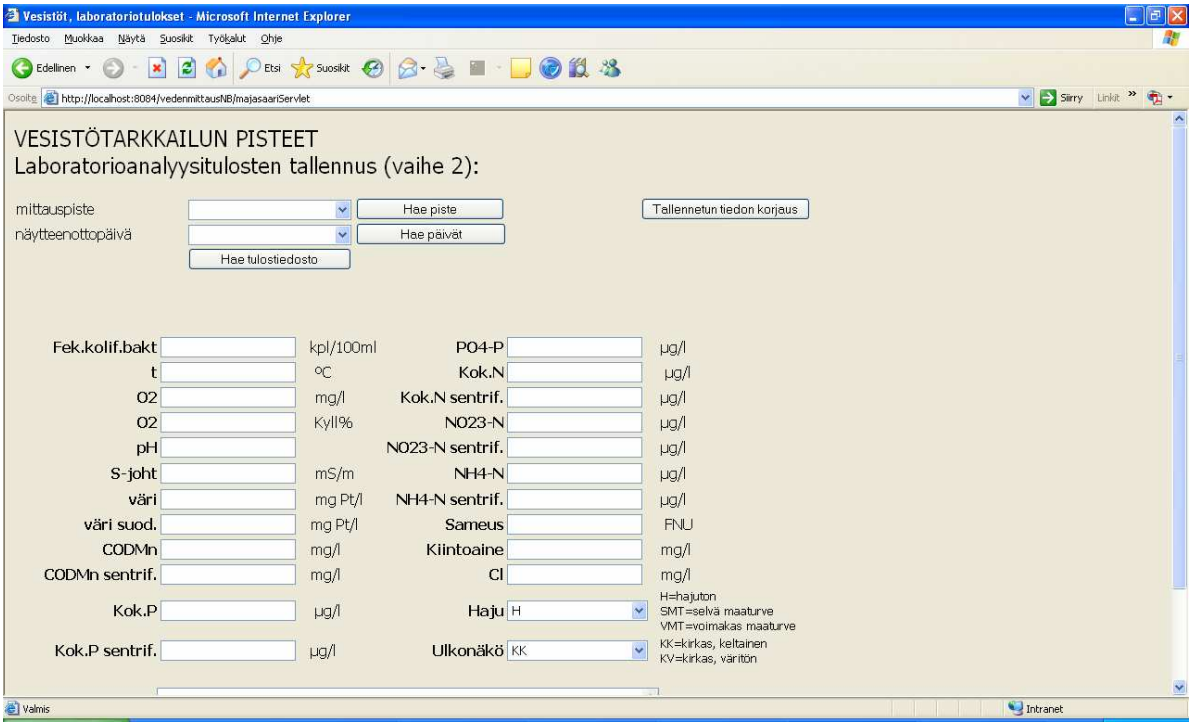
Kuvio 13. Pohjavesi- ja vesistöpisteiden tallennus- ja raportointivalikko

Tietojen tallennusosiossa jätekeskuksen henkilöstö kirjaa mittauspisteistä otetut näytteet. Näytteenottaja tallentaa näytteenottopäivän, näytteenottajan ja tiettyjä vedenmittaukseen liittyviä määreitä. Käyttäjä ei voi muuttaa aiemmin tallentamiaan tietoja. Kenttätietojen tallennus on mittaustulosten tallentamisen ensimmäinen vaihe. (Kuvio 14.)

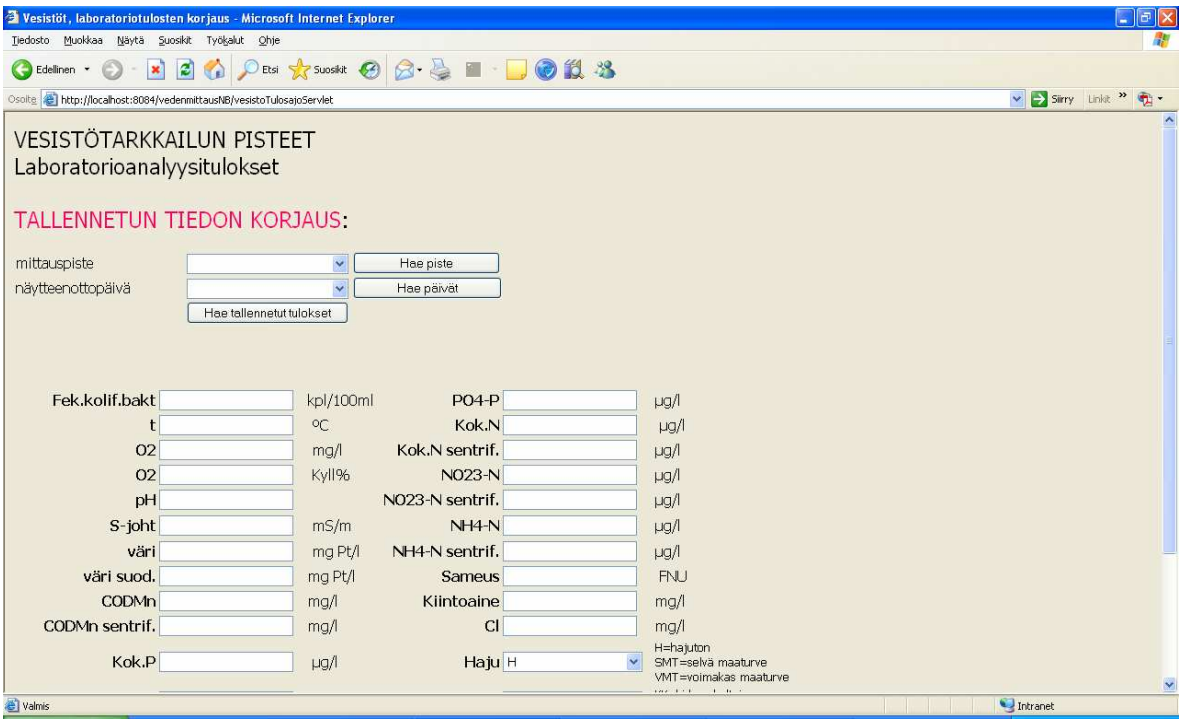


Kuvio 14. Kenttätietojen tallennus -käyttöliittymä

Mittaustulosten tallentamisen toisessa vaiheessa hallinnon henkilöstö kirjaa laboratoriotulokset niistä näytteenotoista, jotka on tallennettu tietokantaan (Kuvio 15.). Tallennettuja laboratoriotuloksia voidaan tarvittaessa korjata erillisen käyttöliittymän avulla (Kuvio 16.). Kaikki sovelluksen käyttäjät voivat tallentamisen jälkeen tulostaa selaimeensa mittaustuloksista viivakaavion ja tulostaulukon valitsemastaan mittauspisteestä, valitsemaltaan aikaväliltä ja valitsemistaan näytetyypeistä.



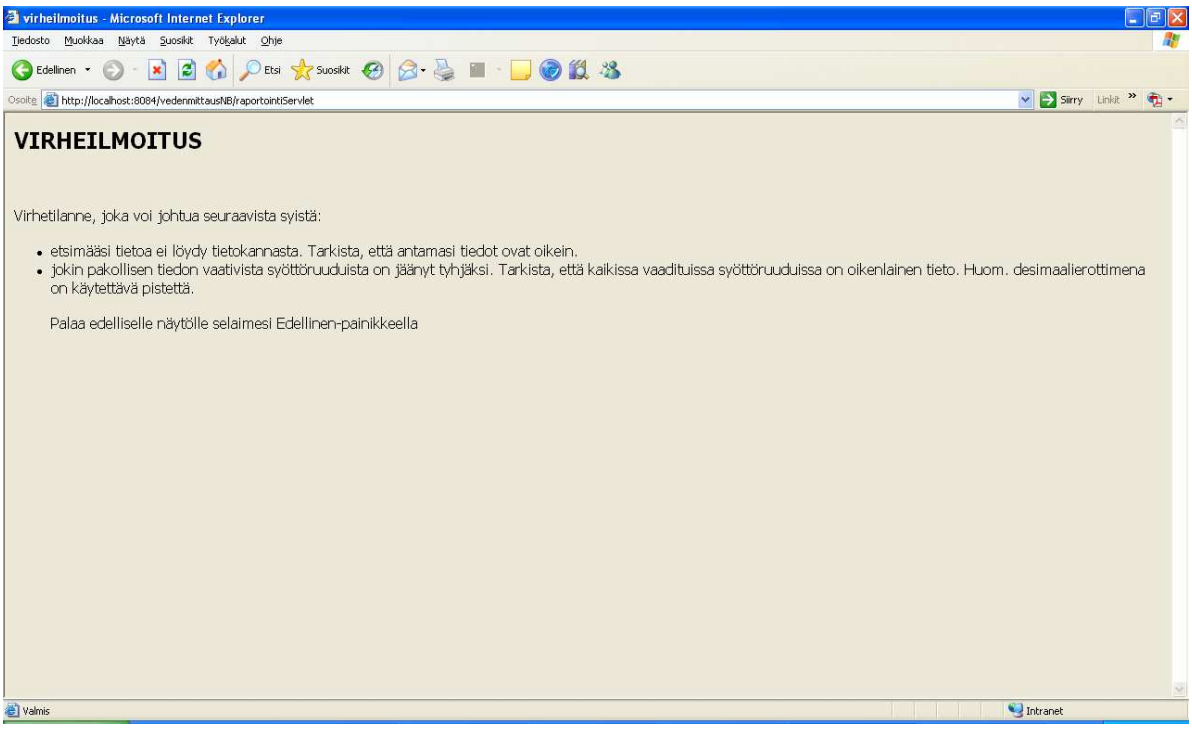
Kuvio 15. Laboratoriotulosten tallentaminen –käyttöliittymä



Kuvio 16. Laboratoriotulosten korjaus –käyttöliittymä

Käyttöliittymiä ja niissä etenemistä on kuvattu laaditussa käyttöliittymäkartassa (Liite 2.). Käyttäjän pääsyä sovelluksen eri osiin on rajattu käyttäjäoikeuksilla. Käyttöliittymissä käyttäjä antaa ohjelmalle tietoja tekemällä valintoja pudotusvalikoiden ja valintapainikkeiden avulla.

Osa tietokannan kyselyistä vaatii, että niihin on oltava annettuna käyttöliittymän avulla jokin arvo; teksti- tai valintaruutu ei saa olla tyhjä tai siinä ei saa olla pudotusvalikon otsikkoriviä. Servleteille on tehty käsittely, joka tarkistaa kriittiset syöttökentät. Mikäli näissä ei ole arvoa tai niissä on virheellinen arvo, annetaan käyttäjälle virheilmoitus (Kuvio 17.).

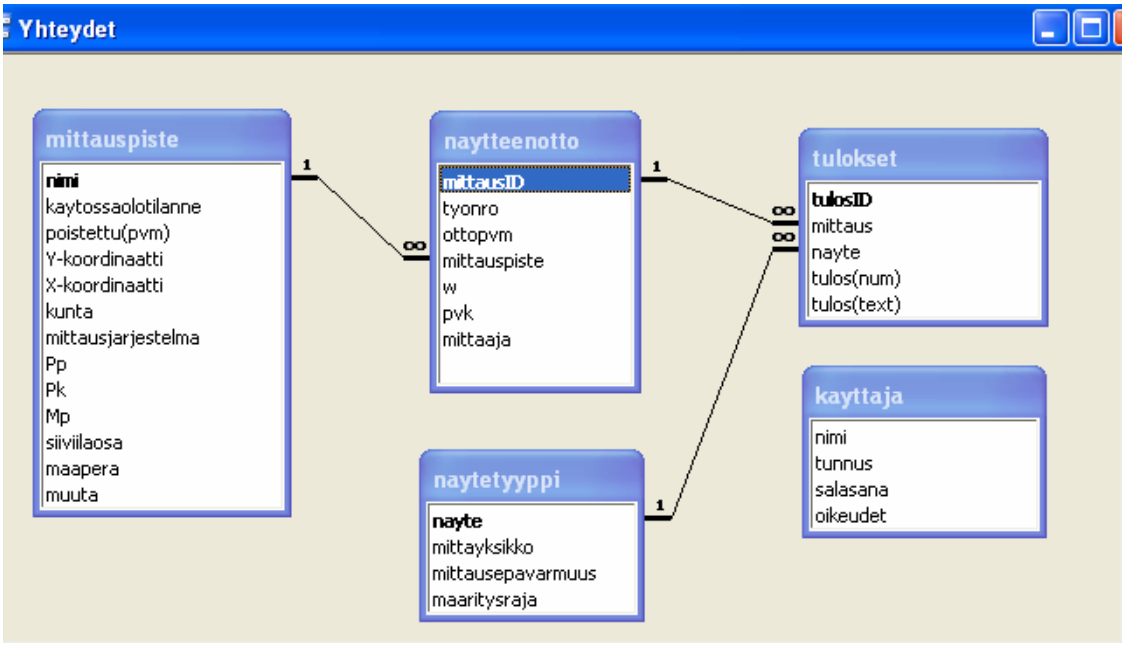


Kuvio 17. Virheilmoitus-käyttöliittymä

3.5 Tietokantamalli

Tietokannaksi valittiin Microsoft Access, koska Eko-Kympillä oli Microsoft Access –ohjelma käytettävissä. Tietokannasta laadittiin ER-kaavio (Liite 3.), jonka avulla tietokanta oli helppo rakentaa. Tietokantaan on laajennettavuuden vuoksi huomioitu kenttiä, joita ei vielä opinnäytetyönä tehdyssä sovelluksessa ole käytetty.

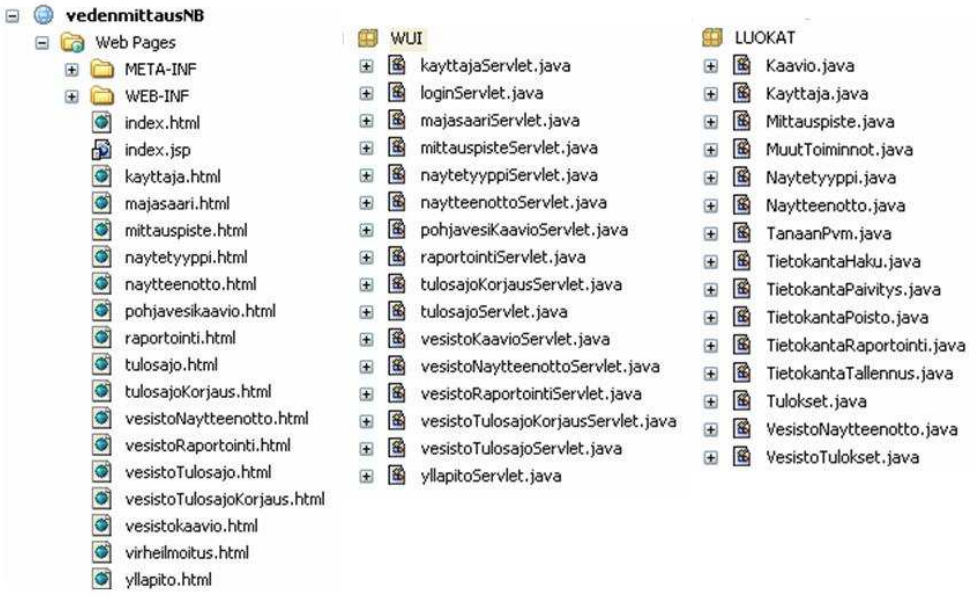
Sovellus käyttää majasaari-nimistä tietokantaa. Se koostuu viidestä taulusta. *Mittauspiste*-taulu sisältää tietoja mittauspisteistä, *naytteenotto*-taulu tietoja suoritetuista näytteiden otoista, *naytetyyppi*-taulu tietoja analyysitulosten määrityksistä, *tulokset*-taulu tietoja vedenmittauksien laboratoriotuloksista ja *kayttajat*-taulu tietoja sovelluksen käyttäjätiedoista. Tietokannan taulujen kentät on määritelty taulukohtaisesti taulukkomuotoon (Liite 4.). Taulukoissa on kuvattu kenttien nimet, kentän sanallinen selite, tietotyyppi, kentän koko merkkeinä ja tieto siitä onko arvo pakollinen. Tietokanta on relaatiotietokanta, joten taulujen välille on määritelty yhteyksiä (Kuvio 18.).



Kuvio 18. majasaari-tietokannan taulujen väliset yhteydet

3.6 Moduulirakenne

Sovelluksen rakenteessa on noudatettu moduulisuunnittelun periaatteita. Sovelluksen luokat jaettiin kahteen päähakemistoon. WUI-hakemistoon tallennettiin kaikki servletit ja LUOKAT-hakemistoon dataluokat. Käyttöliittymät tallennettiin html-muodossa sovelluskehittimen WEB-INF –hakemistoon. (Kuvio 19.) Servletit kutsuvat rajapintojen avulla dataluokkien toimintoja.



Kuvio 19. Sovelluksen moduulirakennemalli

Netbeans muodostaa palvelimelle asennettavaan war-tyyppiseen tiedostoon sovelluskehittimelle määritellyn hakemistorakenteen, joten sovelluksen asennusvaiheessa ei luokkien hakemistorakenteeseeen tarvitse enää puuttua. War-tiedostoa muodostettaessa on kuitenkin varmistettava, että luokat on tallennettu sovelluskehittimelle oikean tyyppisiin hakemistoihin.

3.7 Tietoturvallisuus

Sovellus asennettiin web-palvelimelle ja sitä käytetään Internet-verkon avulla. Tämän vuoksi sovelluksen tietoturvaan kiinnitettiin erityisesti huomiota, jotta vahingonteoilta vältyttäisiin.

Käyttäjille on määritelty kolmenlaisia oikeuksia: lukuoikeudet (0), luku- ja kirjoitusoikeudet (1) ja ylläpitäjän oikeudet (2). Oikeuksien perusteella rajoitetaan käyttäjän pääsyä sovelluksen eri osiin. Lukuoikeudelliset käyttäjät voivat ainoastaan selata sovelluksen raportteja. Tähään ryhmään kuuluvat esimerkiksi ympäristönvalvonnan viranomaiset. Luku- ja kirjoitusoikeudet omaava käyttäjä voi selailun lisäksi tallentaa tietokantoihin tietoja näytteenotoista ja laboratoriotuloksista. Ylläpitäjän oikeuden omaava käyttäjä voi edellä mainittujen toimintojen lisäksi tallentaa käyttäjiä, mittauspisteitä, näytetyyppejä sekä tallentaa ja muokata mittaustuloksia.

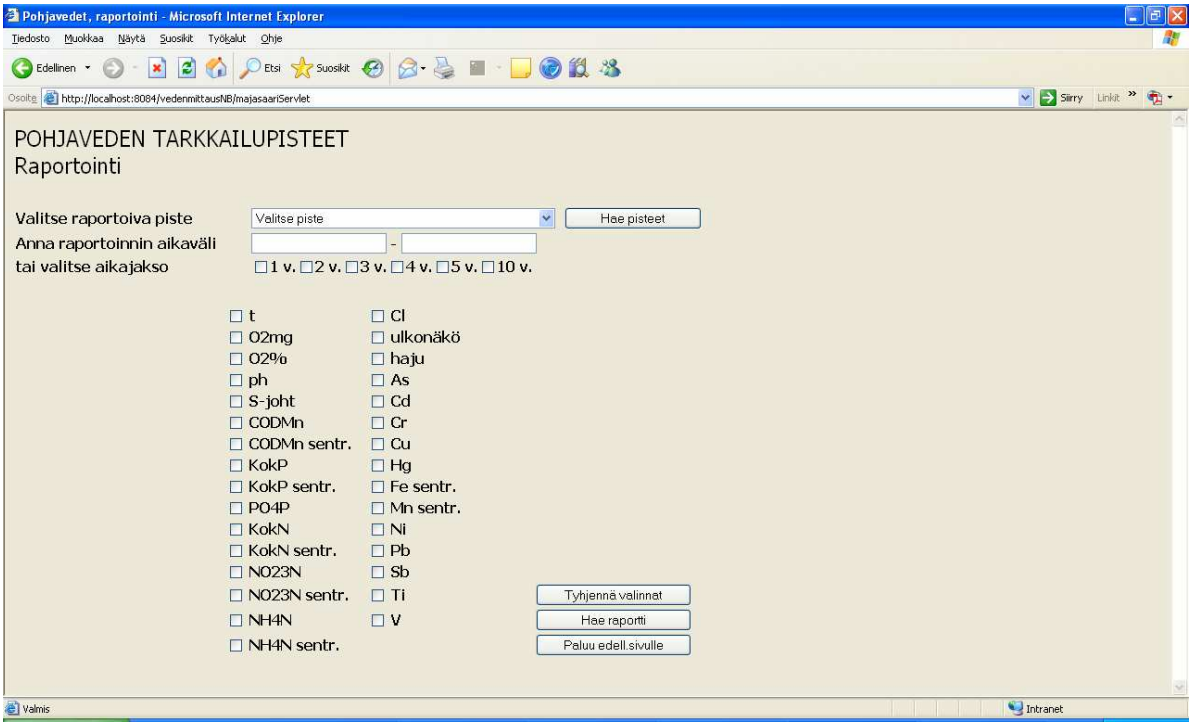
Sovellusistunto alkaa index-sivulta, johon käyttäjän on kirjoitettava käyttäjätunnuksensa ja salasanansa. Näiden perusteella luetaan tietokannasta käyttäjälle annetun oikeuden tunnus, 0, 1 tai 2. Tunnus tallennetaan sovellusistunnon ajaksi ServletContext-olioon, jota kutsutaan ensimmäisenä jokaisen servletin käynnistyessä. Tuolloin tarkistetaan, että ServletContext-oliossa on tallennettuna jokin arvo. Mikäli tallennettua arvoa ei ole määritelty, servlet palauttaa käyttäjän index-sivulle. Tällä voidaan estää tunkeutumiset sovellukseen ilman käyttäjätunnistusta.

3.8 Graafinen kaavio ja taulukko mittaustuloksista

Internetistä saa maksuttomasti JFreeChart-kaaviokomponentin, jonka avulla voi esittää graafisia kaavioita erilaisissa java-sovelluksissa. Komponentti asennetaan sovelluskehittimelle kirjastopakettina, joka sisältää jar-tiedostoja kaavion toteuttamiseksi. Netbeans-

sovelluskehittimelle JFreeChartin kirjasto on helppo asentaa. Asennuksessa on kuitenkin tiedettävä tarkalleen, mihin hakemistoon kirjastopaketti ja erikseen sijoitettavat jar-tiedostot on sijoitettava. Asennuksen jälkeen komponenttia voidaan kutsua normaalin kirjaston tapaan ohjelmakoodiin. JFreeChart-kaaviokomponentin avulla voi tulostaa piirakka-, alue-, pylväs- ja viivakaavioita. Kaaviokomponentti tarvitsee erilaisia määrittelytietoja, jotta komponentti voidaan esittää graafisessa muodossa. Määrittelytiedot annetaan komponentille parametreina. JFreeChart-kaaviokomponentin tuottaman graafisen kaavion voi tallentaa kuvaksi esimerkiksi jpg- tai png-formaattiin (JFreeChart).

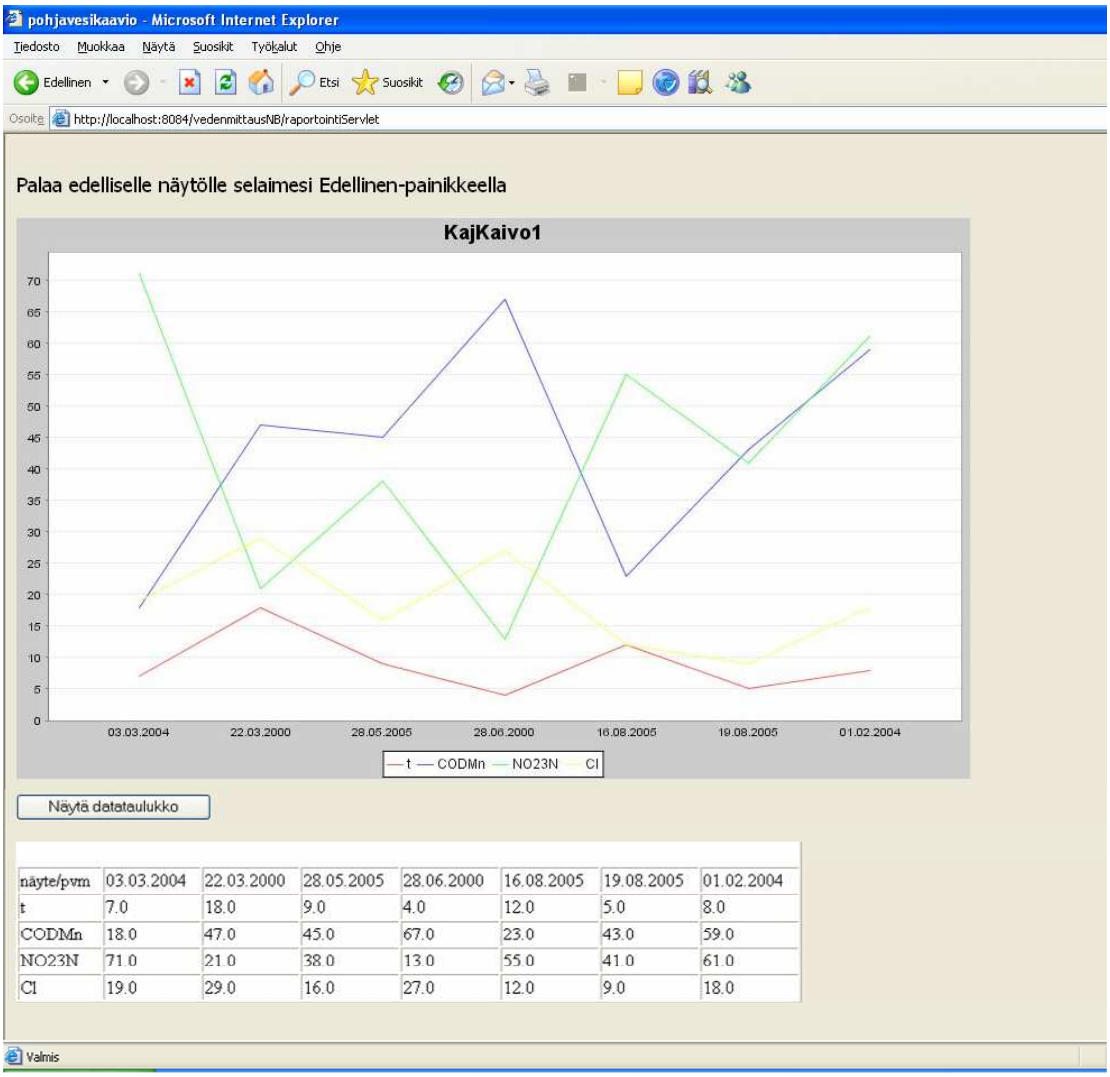
Opinäytetyönä tehdyssä web-sovelluksessa on käytetty JFreeChart-kaaviokomponentin Line Chart -viivakaaviota mittaustulosten visualisoimiseksi. Käyttäjä antaa käyttöliittymän avulla tiedot halutusta mittauspisteestä, mittausten aikavälistä ja valitsee kaaviossa esitettävät näytetyypit (Kuvio 20.). Nämä tiedot välitetään dataluokille servlettien avulla. Luokat kutsuvat metodiensa avulla kaavioon sijoitettavia tietoja tietokannasta. Tietokannasta saadut tiedot tallennetaan taulukkomuuttujiin, joiden avulla tieto siirretään Kaavio-dataluokalle, joka tallentaa kaavion JPG-kuvaksi palvelimen muistiin. Kaavio siirretään JPG-kuvana hmtl-koodin joukossa sitä kutsuvalle käyttöliittymälle. Kaavion toiminnallisuutta kuvataan seksenssikaaviossa (Liite 5.).



Kuvio 20. Kaavion lähtötiedot -käyttöliittymä

Viivakaaviossa esitetään käyttäjän valitsemien näytteiden mittaustulokset halutulta ajalta. Kaavion x-akselille tulostuvat näytteiden ottopäivät, otsikoksi mittauspisteen nimi ja y-askelille mittaustuloksia sisältävästä taulukkomuuttujasta saadut ääreisarvot. Jokaista valittua näytettä vastaa yksi erivärinen viiva kaaviossa.

Ohjelman suorituksen aikana kaaviokomponentin tarvitsemat määrittelytiedot tallennettiin erilaisiin taulukko-muuttujiin, joiden avulla määrittelytietoparametreista saatiin dynaamisia. Kaavion tarvitsemat parametrit tallennetaan servletissä myös sovellusistunnon aikaiseen ServletContext -olioon. Näitä tietoja käytetään kaavion tietojen esittämiseksi käyttäjälle taulukkomuodossa. Servlet Context -olion välityksellä määrittelytiedot siirretään sille servletille, joka siirtää kaavion näytölle tulostettavaksi. Servlet esittää käyttäjälle html-sivun, jonka näytä datatataulukko -painiketta painamalla luetaan ServletContext-olioon tallennetut tiedot ja muodostetaan niistä taulukko html-sivulle kaavion alapuolelle. (Kuvio 21.).



Kuvio 21. Kaavio ja taulukko tulostettuna näytölle

4 POHDINTA

Opinnäyte oli haastava työ. Eko-Kympillä ei ole ohjelmoinnin asiantuntijaa, joten apua ohjelmoinnin ongelmiin piti etsiä Internetistä, kirjallisista lähteistä ja kehitysympäristön ongelmissa myös ohjelmoinnin opettajilta. Sovelluskehitys aloitettiin huolellisella kartoituksella ja suunnittelulla, sillä tarkan vaatimusmäärittelyn avulla ohjelman rakenteen moduloinnti oli helpompaa. Ne kohdat, jotka olivat vaikeampia ja vaativat asioiden selvittämistä, voitiin jättää odottamaan ratkaisua eikä se hidastanut muiden osien eteenpäin viemistä. Arkkitehtuurisuunnittelun merkitys kirkastui tällaisessa tilanteessa.

Eko-Kympillä oli selkeä näkemys ohjelman toiminnallisuudesta. Tästä syystä sovelluskehitystä voitiin tehdä niin, että toiminnallisiin vaatimuksiin tuli vain vähäisiä muutoksia sovelluskehityksen aikana. Vaatimusmäärittely onnistui erittäin hyvin. Se helpotti valtavasti ohjelmointityöhön keskittymistä.

Opinnäyte oli hyvin opettavainen. Ilman asiantuntija-apua ohjelmointityö oli ajoittain hidasta. Ratkaisuja erilaisiin ohjelmointiongelmiin löytyi pääasiassa Internetistä. Sitkeällä hakemisella ja opiskelemisella vaikeatkin ongelmat lopulta selvisivät. Oppia tuli ”kantapään kautta”. Sen lisäksi englannin kielen taitoni kehittyi valtavasti ja opin käyttämään Internetiä tehokkaasti ohjelmoijan tietolähteenä.

Vaikein osa ohjelmoinnissa oli dynaamisen kaavion esittäminen näytöllä. Pienen Internetistä saadun vihjeen perusteella löytyi JFreeChart-sivusto, josta kirjastopaketin lataamalla saatiin tarpeelliset jar-tiedostot toimintojen tekemiseen. Kaaviokomponentti on todella erinomainen väline visuaalisen tulosten esittämiseksi. Kaavion ohjelmointi oli aluksi selkeiden ohjeiden ja koodiesimerkkien puuttuessa hieman työlästä. Muutamien Internetistä saatujen opastuksien perusteella komponentin toiminnot kuitenkin selkiytyivät. Kuten usein ohjel-

moinnissa, tässäkin piti paikkansa totuus, että kun asia alkaa tuntua liian monimutkaiselta, on siihen varmasti olemassa yksinkertainen ratkaisu. Tällöin on viisainta aloittaa alusta ja lähteä ratkaisemaan ongelmaa uudella tavalla.

Ohjelman määrittelyvaiheessa käyttäjiksi kuvattiin Eko-Kympin jätekeskuksen ja hallinnon henkilöstö sekä ympäristövalvonnan viranomaiset. Sovellus vastasi hyvin näiden käyttäjien tarpeita. Eko-Kymppi haluaa tiedottaa avoimesti jätekeskukseen liittyvistä asioista ja sovelluskehityksen loppuvaiheessa esitettiinkin, että sovelluksen käyttäjäkuntaa laajennettaisiin. Eko-Kymppi haluaa liittää sovelluksen kotisivuilleen, josta se on kaikkien Internetin käyttäjien saatavilla. Tämä vuoksi sovellukseen on tehtävä muutamia lisäyksiä esimerkiksi ohjelman käyttötarkoituksen kuvaamiseksi ja käyttöohjeiden selventämiseksi. Sovittiin, että edetään jonkin aikaa alkuperäisen suunnitelman mukaisesti. Myöhemmin, kun sovelluksen toiminnasta on saatu enemmän kokemusta, päivitetään sovellus ja avataan sen käyttö laajemmalle käyttäjäkunnalle.

Eko-Kymppi on mukana jätekeskuksien kaasujen mittauksiin liittyvässä SIMIRA-projektissa. On käyty neuvotteluja siitä, että opinnäytteenä toteutetusta sovelluksesta tehdään laajempi versio, johon voidaan liittää Eko-Kympin SIMIRA-projektiin liittyviä mittauksia ja toimintoja. Opinnäytteenä tehdyn sovelluksen lähtökohtana oli, että siitä tehdään rakenteellisesti sellainen, että sen laajentaminen on helppoa. On erittäin haasteellista ja palkitsevaa ryhtyä jatkokehitystyöhön.

LÄHTEET

EHP-Tekniikka, 2006. Mittausjärjestelmät, tuotteet ja palvelut. EHP-Tekniikka. http://www.ehp-tekniikka.fi/palvelut.html (Luettu 9.4.2006)

Eko-Kymppi, 2006, 1a. Kotisivut. Kainuun jätehuollon kuntayhtymä Eko-Kymppi. http://www.eko-kymppi.fi/aloitusnakyma.htm (Luettu 9.4.2006)

Eko-Kymppi, 2006, 1b. Majasaarenkankaan jätekeskuksen ympäristölupa. Kainuun jätehuollon kuntayhtymä Eko-Kymppi. http://www.eko-kymppi.fi/majasaari/Majasaari%20KAI%20lupa.htm#_Toc13986484 (Luettu 9.4.2006)

Hatakka Taina, 2004. Suorituskykyisten web-sovellusten määrittely ja suunnittelu. http://www.cs.helsinki.fi/u/verkamo/sem/gradu_k2005/hatakka2.pdf , sivu 5 (Luettu 8.4.2006)

Haikala Ilkka & Märijärvi Jukka, 2002. Ohjelmistotuotanto. Korkeakoulu/Satku.fi -sarja. Helsinki: Talentum Media Oy.

Helsingin yliopisto, 2002. Ohjelmistoarkkitehtuurit, luentomateriaalia. Helsingin yliopisto, Tietojenkäsittelytieteen laitos. http://www.cs.helsinki.fi/u/aptuovin/slides-08-03-2002.pdf (Luettu 27.4.2006)

Helsingin yliopisto, 2005. Arkkitehtuuri, prosessi ja organisaatio, luentomateriaalia. Helsingin yliopisto, Tietojenkäsittelytieteen laitos. http://www.cs.helsinki.fi/u/viljamaa/oa-k2005/slides-02.pdf (Luettu 27.4.2006)

Helsingin yliopisto, 2006 1a. Ohjelmistoarkkitehtuurit, luentomateriaalia. Helsingin yliopisto, Tietojenkäsittelytieteen laitos. http://www.cs.helsinki.fi/u/viljamaa/oa-k2005/slides-01.pdf (Luettu 26.4.2006)

Helsingin yliopisto, 2006 1b. Arkkitehtuurityylit, luentomateriaalia. Helsingin yliopisto, Tietojenkäsittelytieteen laitos.
http://www.cs.helsinki.fi/u/jviljama/arkkitehtuurit/k06/oa-08-arkkitehtuurityylit-a.pdf (Luettu 26.4.2006)

Nakhimovsky Alexander & Myers Tom. 2002. Java & XML. IT-Press/Inside-sarja. Helsinki: Edita Publishing Oy.

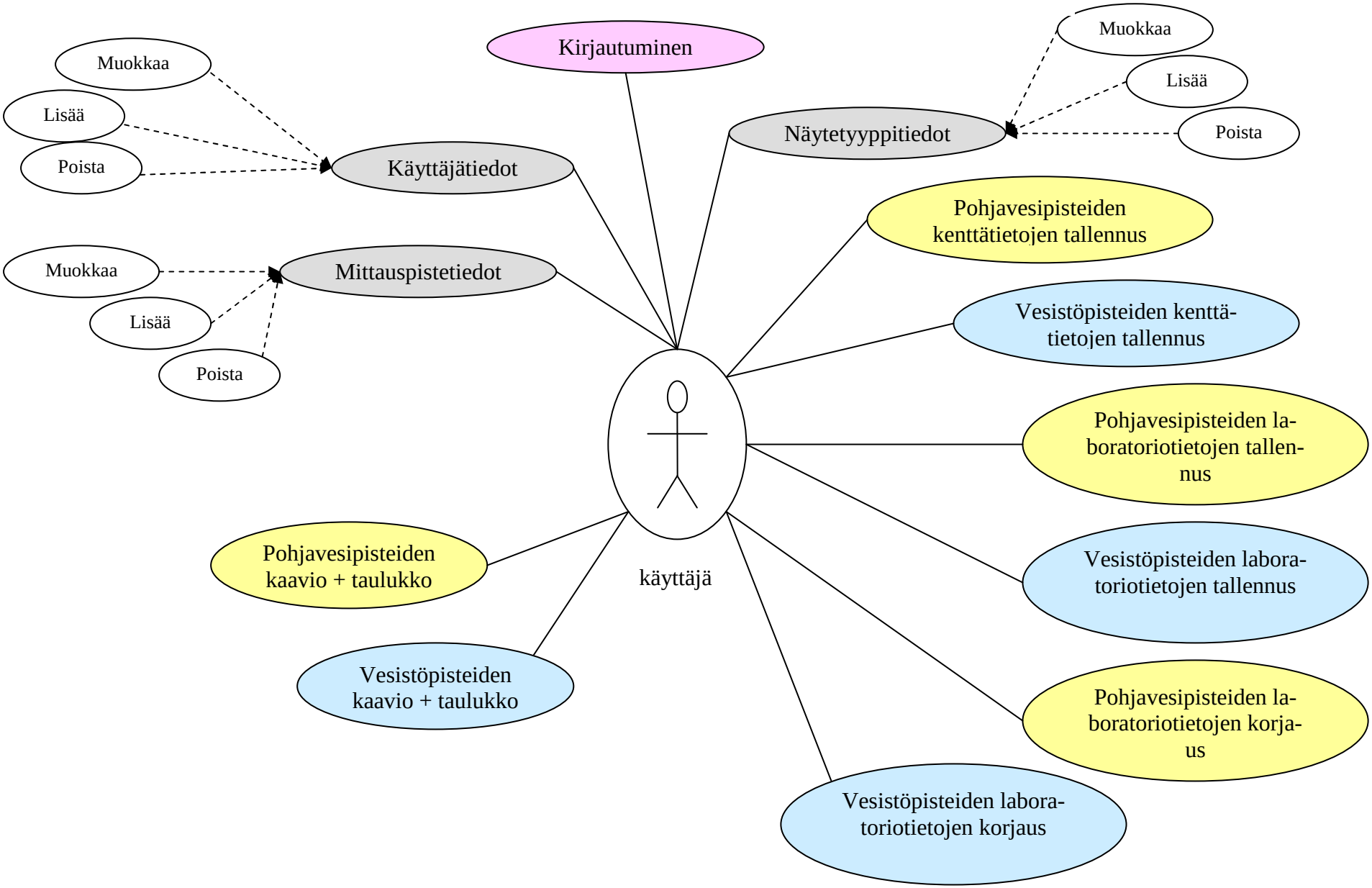
Turun yliopisto, 2002.Arkkitehtuurisuunnittelu, luentomateriaalia. Turun yliopisto, Informaatioteknologian laitos.
http://staff.cs.utu.fi/kurssit/ohjelmistotuotanto/syksy_2002/luennot02/Arkkitehtsuun(12).pdf (Luettu 20.4.2006)

LIITTEET

Sovellusta kuvaavat kaaviot ja taulukot

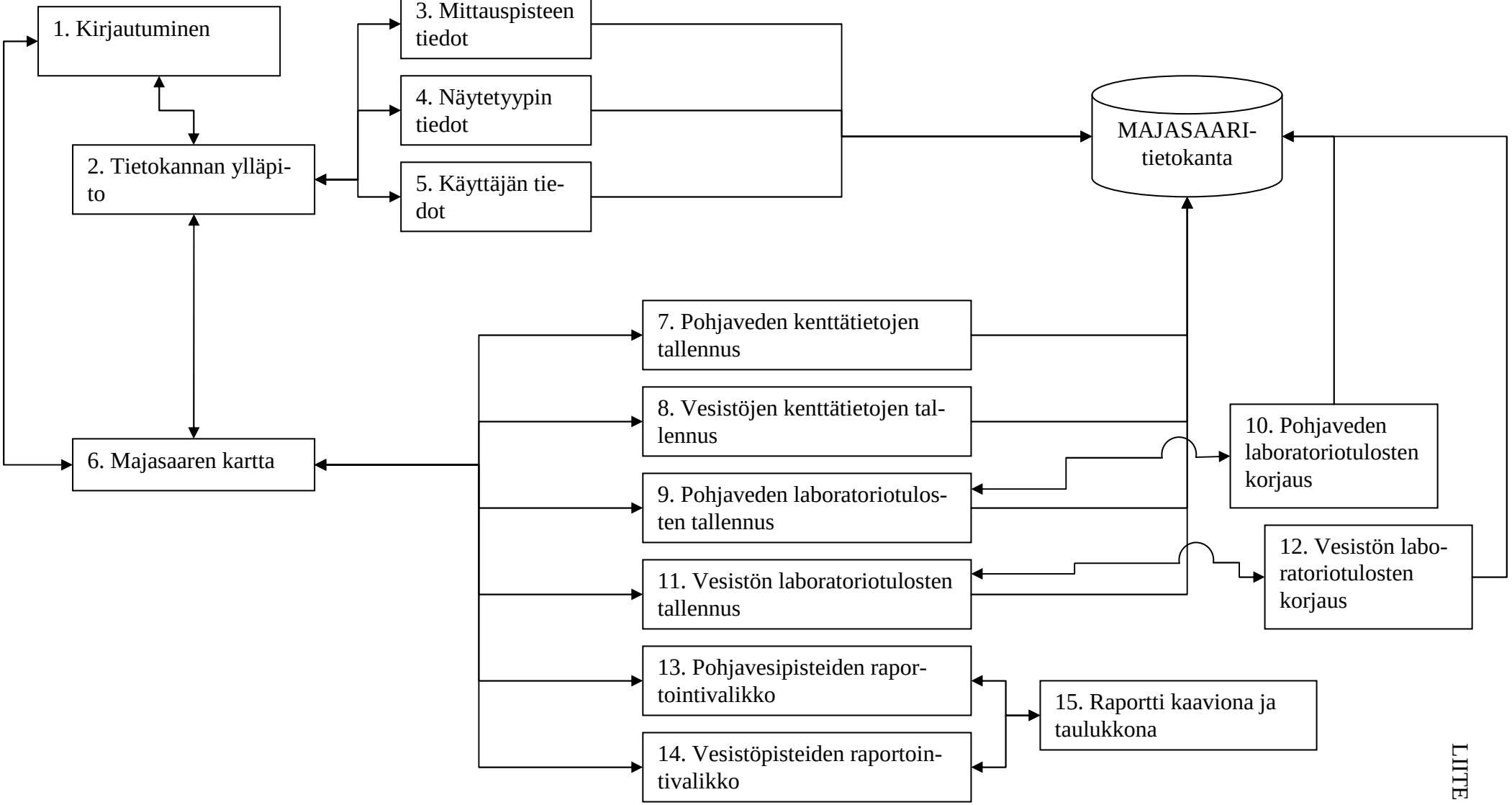
Liite 1: käyttötapauskaavio
Liite 2: käyttöliittymäkartta
Liite 3: tietokannan ER-kaavio
Liite 4: tietokannan taulut ja niiden kentät
Liite 5: viivakaavion ja taulukon sekvenssikaavio

KÄYTTÖTAPAUSKAAVIO



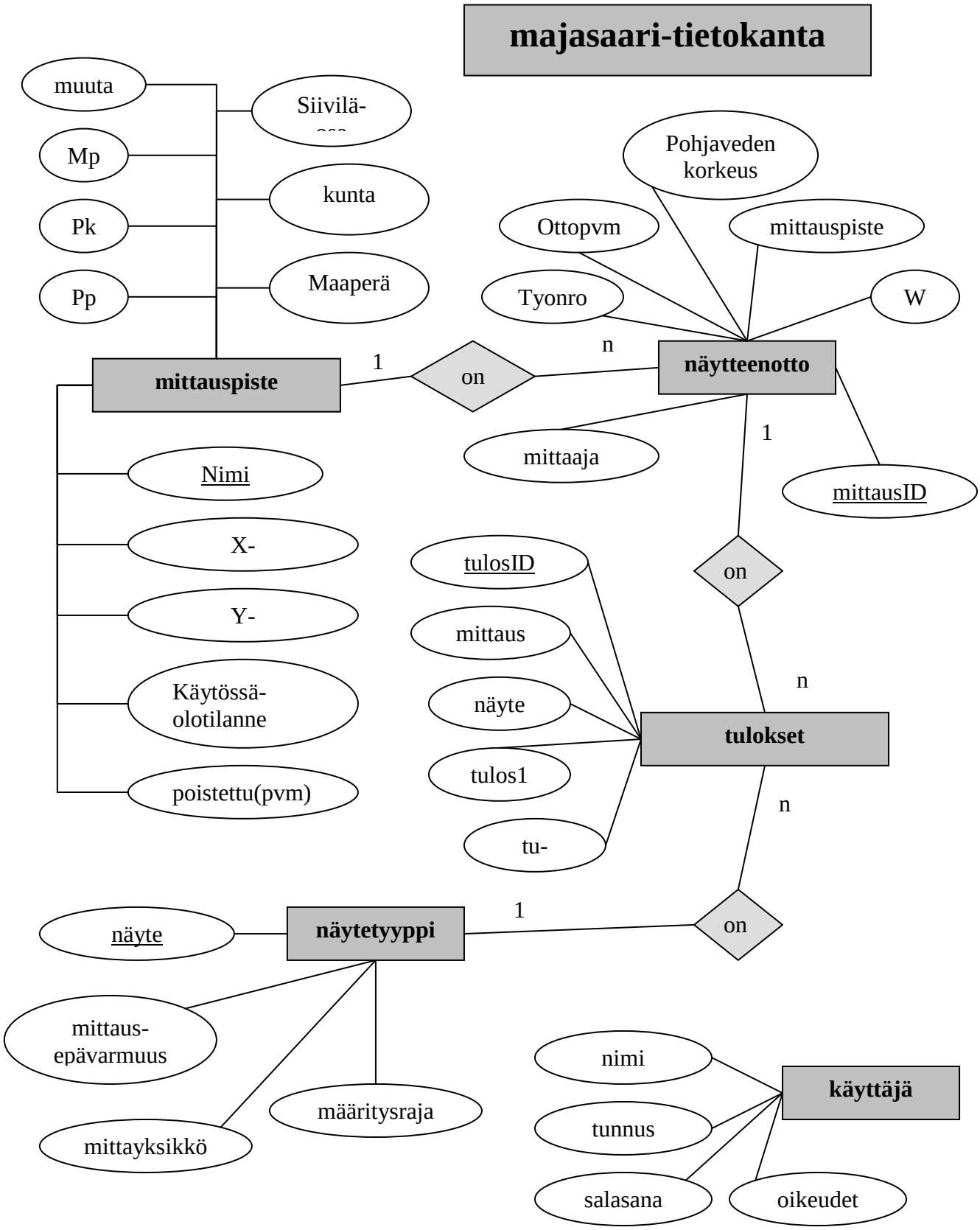
LIITE 1.

KÄYTTÖLIITTYMÄKARTTA



LIITE 2.

ER-KAAVIO


majasaari-tietokanta
muuta
Siivilä-
Mp
kunta
Pk
Maaperä
Pp
Pohjaveden korkeus
Ottopvm
mittauspiste
Tyonro
W
mittauspiste
1
on
n
näytteenotto
mittaaja
1
mittausID
Nimi
X-
Y-
Käytössä-olotilanne
poistettu(pvm)
tulosID
mittaus
näyte
tulos1
tu-
on
n
tulokset
n
näyte
näytetyyppi
1
on
mittaus-epävarmuus
määritysraja
mittayksikkö
nimi
käyttäjä
tunnus
salasana
oikeudet

TIETOKANNAN TAULUT JA NIIDEN KENTÄT

mittauspiste –taulu

kentän nimi	selite	tietotyyppi	kentän koko	arvo vaaditaan
nimi	primary key, mittauspisteen nimi	teksti	25	kyllä
Y-koordinaatti	sijaintikoordinaatti,Y	teksti	10	kyllä
X-koordinaatti	sijaintikoordinaatti, X	teksti	10	kyllä
kunta	mittauspisteen sijaintikunta	teksti	15	kyllä, oletus Kajaani,
mittausjarjestelma	käytettävä mittausjärjestelmä	teksti	3	kyllä, oletus N60,
Pp	putken yläpää	luku	-	kyllä
Pk	putken kärki	luku	-	kyllä
Mp	maanpinta	luku	-	kyllä
maapera	maaperän pohjatietoja	teksti		ei
siivilaosa	siiviläosa metreinä	luku	-	kyllä
kaytossaolotilanne	onko mittauspiste käytössä	luku	-	kyllä, oletus 1 1=käytössä 0=ei käytössä
poistettu(pvm)	poistettu käytöstä pvm	päivämäärä	-	ei
muuta	sekalaista tietoa	teksti	100	ei

naytteenotto-taulu

kentän nimi	selite	tietotyyppi	kentän koko	arvo vaaditaan
mittausID	primary key	laskuri		
työnro	työnumero	luku	-	ei
ottopvm	mittauksen ottopäivä	päivämäärä	-	kyllä
mittauspiste	secondary key, mittauspisteen nimi	teksti	25	kyllä
w	vesipinta	luku	-	ei
pvk	funktio (Pp-W)	luku	-	ei
mittaaja	mittauksen suorittajan nimi	teksti	30	kyllä

naytetyyppi-taulu

kentän nimi	selite	tietotyyppi	kentän koko	arvo vaaditaan
näyte	primary key, näytteen nimi	teksti	11	kyllä
mittayksikkö	näytteen mittayksikkö	teksti	6	ei
mittausepävarmuus	mittauksen epävarmuus%	luku	-	ei
määritysraja	mittaustuloksen alaraja	luku	-	ei

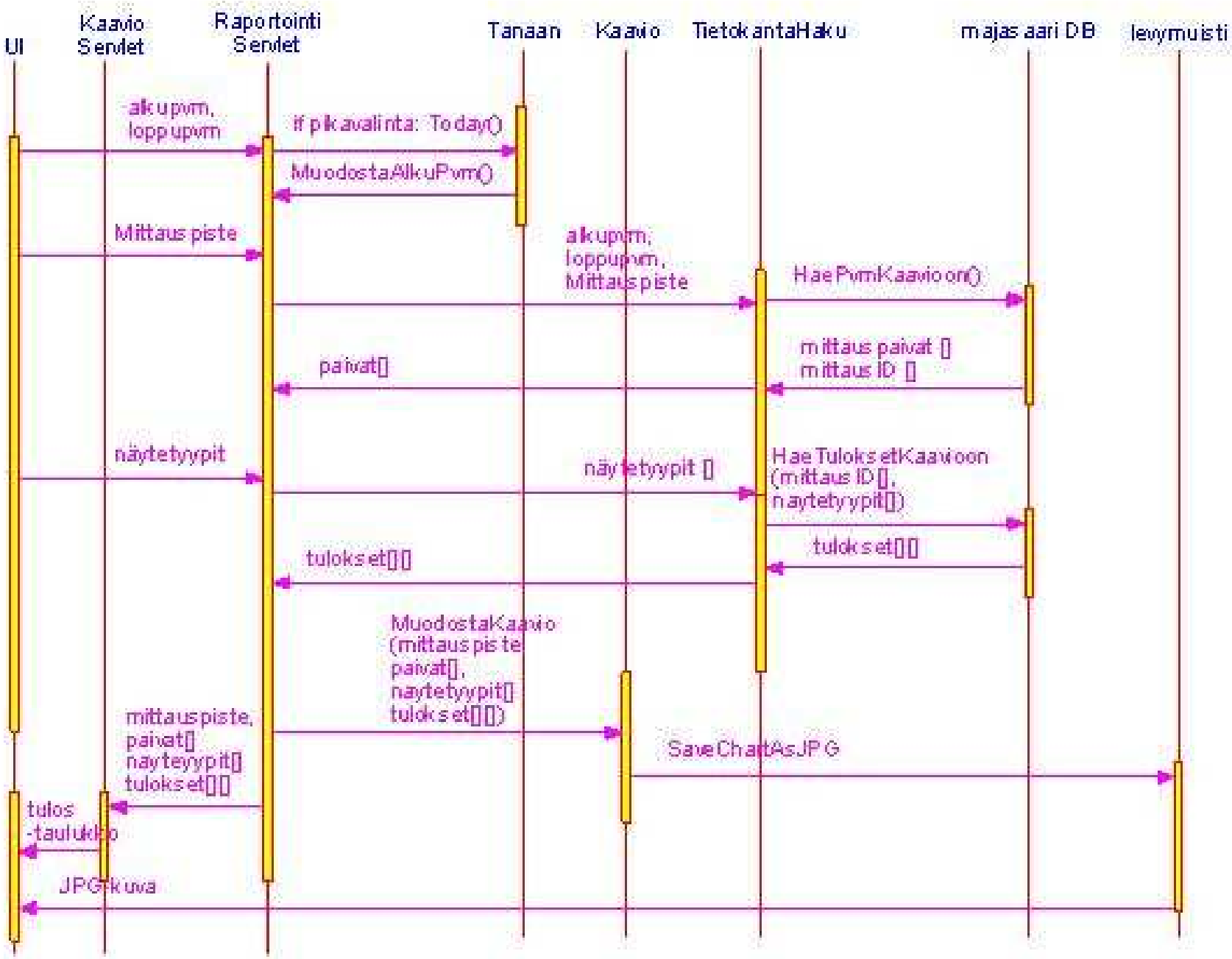
tulokset-taulu

kentän nimi	selite	tietotyyppi	kentän koko	arvo vaaditaan
tulosID	primary key	laskuri	-	kyllä
mittaus	secondary key, mittausID	luku	-	kyllä
näyte	secondary key, näytteen nimi	teksti	11	kyllä
tulos(num)	mittauksen numeerinen tulos	luku	-	ei
tulos(text)	mittauksen sanallinen tulos	teksti	10	ei

kayttaja-taulu

kentän nimi	selite	tietotyyppi	kentän koko	arvo vaaditaan
nimi	käyttäjän nimi	teksti	50	ei
tunnus	käyttäjätunnus	teksti, Unique (tunnus, salasana)	10	ei
salasana	käyttäjän salasana	teksti, Unique (tunnus, salasana)	10	ei
oikeudet	käyttäjän saamat käyttöoikeudet	teksti	50	ei

SEKVENSSIKAAVIO



LIITE 5.